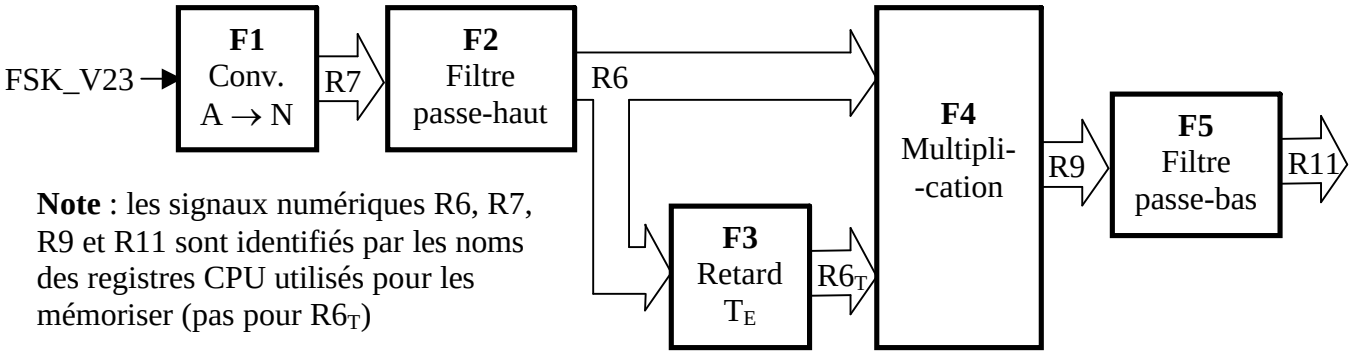
CLIP V1 & V2

Démodulation FSK V23 numérique sur MSP430

Cette étude est basée sur le document SLAA037.PDF disponible sur le site de Texas Instruments : ti.com.

1. Schéma fonctionnel

On utilise ici un procédé proche de la démodulation de fréquence en quadrature utilisé couramment en analogique. La fonction "déphasage" devient ici une fonction "retard".

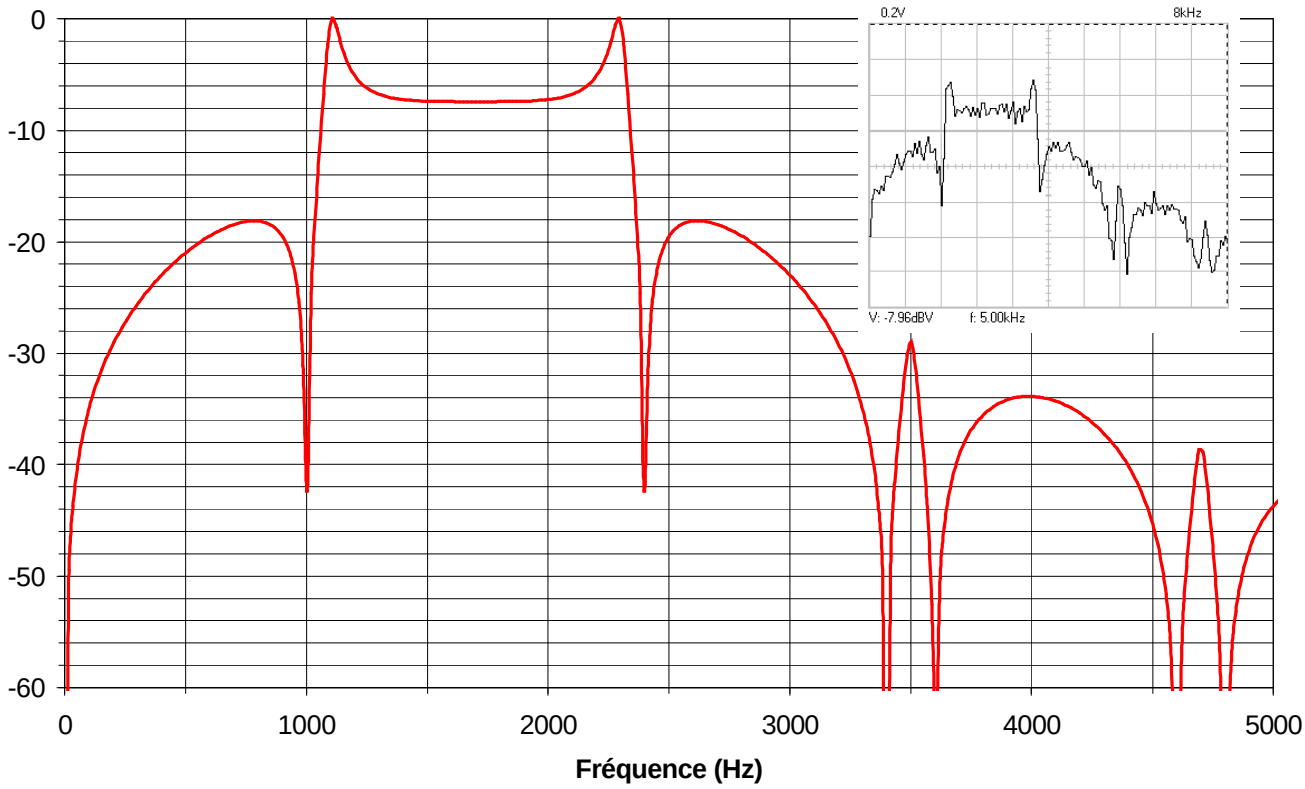


Note : les signaux numériques R6, R7, R9 et R11 sont identifiés par les noms des registres CPU utilisés pour les mémoriser (pas pour $R6_T$)

Le signal FSK_V23 est le signal analogique modulé à traiter. Il est conforme aux recommandations V23, c'est à dire :

- Fréquence centrale : F_0 = 1700 Hz
- Fréquence basse F_Z = 1300 Hz
- Fréquence haute F_A = 2100 Hz
- Rapidité de modulation : 1200 bauds

Le spectre relatif d'un signal FSK de ce type modulé par une séquence pseudo-aléatoire est le suivant :



On a inséré, en haut à droite, un relevé expérimental.

Toutes les fonctions sont numériques, à l'exception de la fonction mixte F1. Les calculs sont effectués par un programme d'interruption au rythme de la **fréquence d'échantillonnage F_E = 6800 Hz**.
Cette fréquence relativement basse a été choisie pour respecter les critères suivants :

- Règle de Shannon : $F_E > 2$ Fin max. Le spectre du signal FSK montre des composantes non négligeables au-delà de 3400 Hz ($F_E/2$). Un filtrage analogique préalable (avant la conversion A/N) est donc nécessaire.
- Le microcontroleur n'est pas un DSP. Il doit être capable de réaliser tous les calculs à chaque échantillon.
- Faciliter la réalisation de la fonction F3 "retard de $T_E = 1/F_E$"

1.1 F1 : conversion A → N

On utilise la technique du "simple rampe". La partie logicielle est réduite par l'utilisation des fonctions "Timer" intégrées dans chaque MSP430.
Le résultat est codé sur 16 bits en code complément à 2 et placé dans le registre R7 du CPU (voir §2).

$$R7 = K_{AN} \frac{V_{FSK_V23} - V_{OFFSET}}{V_{REF}}$$

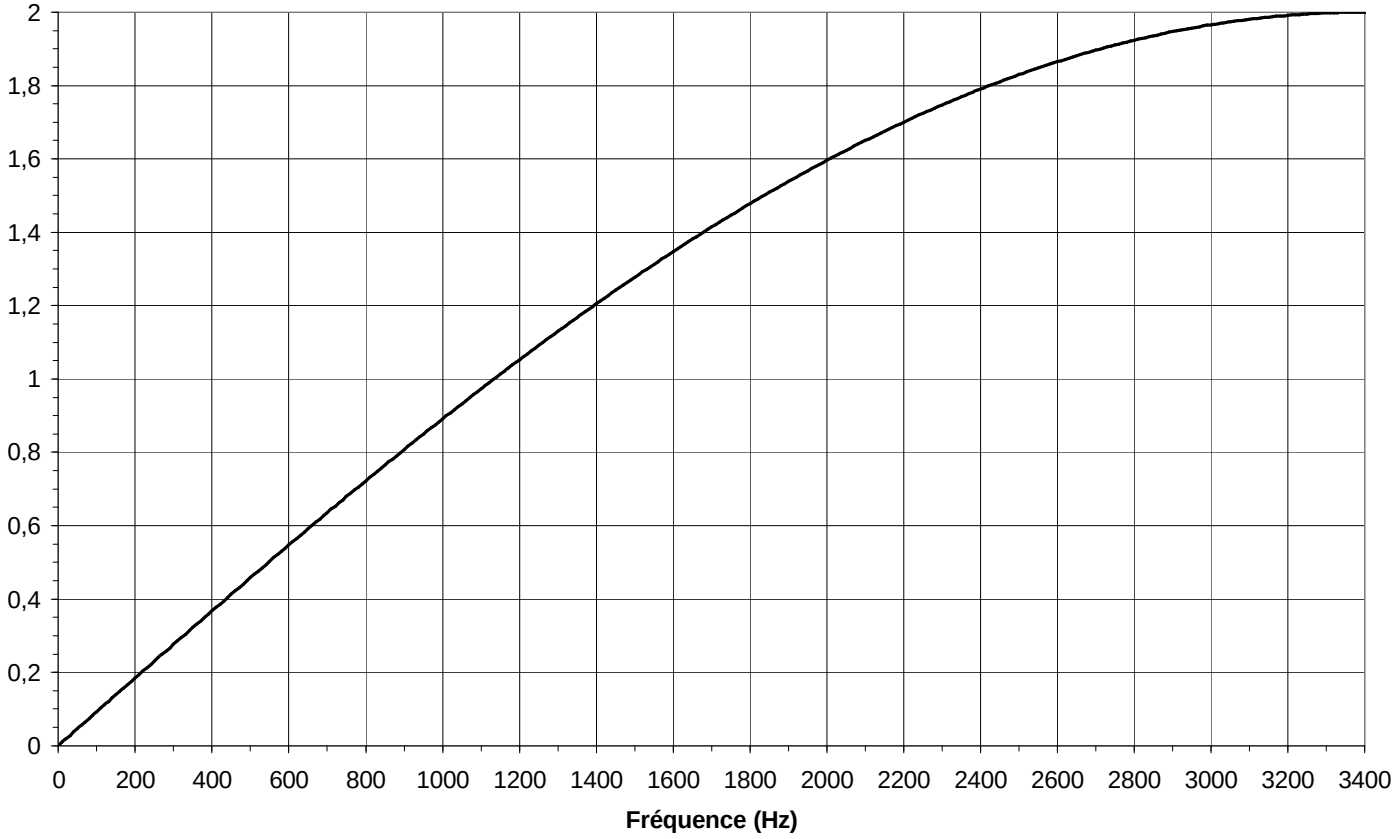
$K_{AN} = 32768$
V_{OFFSET} : constante représentant une tension de décalage
V_{REF} : tension de référence. Dépend de la pente de la rampe et de la fréquence d'horloge

1.2 F2 : Filtre passe-haut

En fait, son action principale est d'éliminer la composante continue de R7 et de rendre ainsi la fonction démodulation relativement insensible à la tension moyenne du signal FSK_V23.

Transformée en Z de la fonction de transfert : $\frac{R6(z)}{R7(z)} = 1 - Z^{-1}$

La réponse en fréquence théorique est la suivante (limitée à $F_E/2$):



1.3 F3 : Retard T_E

Cette fonction reproduit sur $R6_T$ le signal numérique R6 retardé d'une période d'échantillonnage.

$$R6_T = R6 \times Z^{-1}$$

Le module de sa transmittance est toujours égal à 1.

1.4 F4 : Multiplication

Cette fonction réalise, comme son nom l'indique : $R9 = R6 \times R6_T$

Les signaux R9, comme R6 et $R6_T$, sont codés en C2 sur 16 bits. Les valeurs de R6 et $R6_T$ doivent rester suffisamment faibles pour éviter les dépassements.

La valeur moyenne de R9 contient l'information "déviation de fréquence", c'est à dire le résultat de la démodulation de fréquence.

En effet :

- On admet pour simplifier que R6 est un signal analogique sinusoïdal modulé en fréquence autour de F_0 :
  $R6 = E \times \cos(2 \times \pi \times (F_0 + dF) \times t)$ dF étant la déviation de fréquence par rapport à F_P (± 400 Hz)
- On en déduit : $R6_T = E \times \cos(2 \times \pi \times (F_0 + dF) \times (t - T_E))$ avec $T_E = \frac{1}{F_E} = \frac{1}{6800}$ = 147µS = cste
- Or F_E = 4x F_0. Il en résulte : $R6_T = E \times \cos\left(2 \times \pi \times (F_0 + dF) \times t - \frac{\pi}{2} - \frac{\pi}{2} \times \frac{dF}{F_0}\right)$
- La fonction F4 produit alors le signal :

$$R9 = R6 \times R6_T = E^2 \times \cos(2 \times \pi \times (F_0 + dF) \times t) \times \cos\left(2 \times \pi \times (F_0 + dF) \times t - \frac{\pi}{2} - \frac{\pi}{2} \times \frac{dF}{F_0}\right)$$

$$R9 = \frac{E^2}{2}\left[\cos\left(4 \times \pi \times (F_0 + dF) \times t - \frac{\pi}{2} - \frac{\pi}{2} \times \frac{dF}{F_0}\right) + \cos\left(\frac{\pi}{2} + \frac{\pi}{2} \times \frac{dF}{F_0}\right)\right]$$

$$R9 = \frac{E^2}{2}\left[\underbrace{\cos\left(4 \times \pi \times (F_0 + dF) \times t - \frac{\pi}{2} - \frac{\pi}{2} \times \frac{dF}{F_0}\right)}_{\text{Composante de fréquence } 2xF_0 \text{ (3400Hz) environ}} \underbrace{- \sin\left(\frac{\pi}{2} \times \frac{dF}{F_0}\right)}_{\text{Composante utile}}\right]$$

Composante de fréquence $2xF_0$ (3400Hz) environ
Éliminée par le filtre passe-bas F5

Composante utile
Passe au travers du filtre F5 car de fréquence 600Hz (1200 bauds)

1.5 F5 : filtre passe-bas

Sa tâche est d'éliminer la composante de R9 de fréquence $2xF_0$, tout en laissant passer la composante utile (le signal modulant). Voir le § ci-dessus.

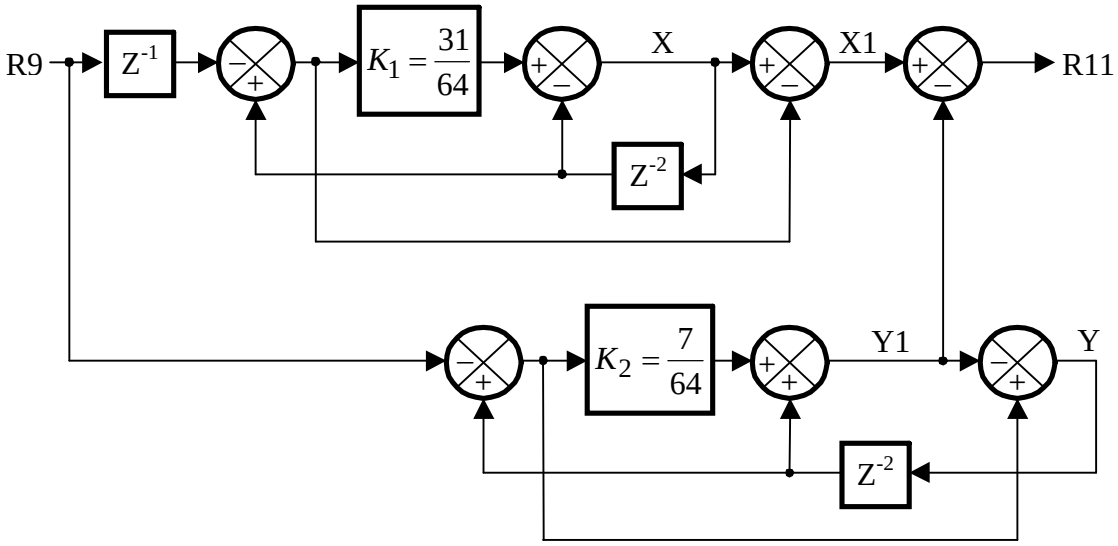
Dans cette application, le signal modulant est binaire de rapidité 1200 bauds. Le signal R9 doit donc permettre de produire un signal binaire RXD suffisamment fidèle pour être traitée par la fonction "réception série asynchrone".

Le filtre F5 est du type IIR et utilise la technique des "ondelettes". Ce type de filtre est stable et est peu sensible à la tolérance des coefficients. Cela a permis d'éviter toute opération de multiplication et ainsi réduire considérablement le temps de calcul (le MSP430 n'est pas un DSP).

Caractéristiques du filtre :

- Type Butterworth d'ordre 5
- Bande passante à –1dB : 1400Hz
- Ondulations dans la bande passante : < 1dB
- Bande coupée à –40dB : 2800Hz

Schéma opérationnel :



Fonctions de transfert :

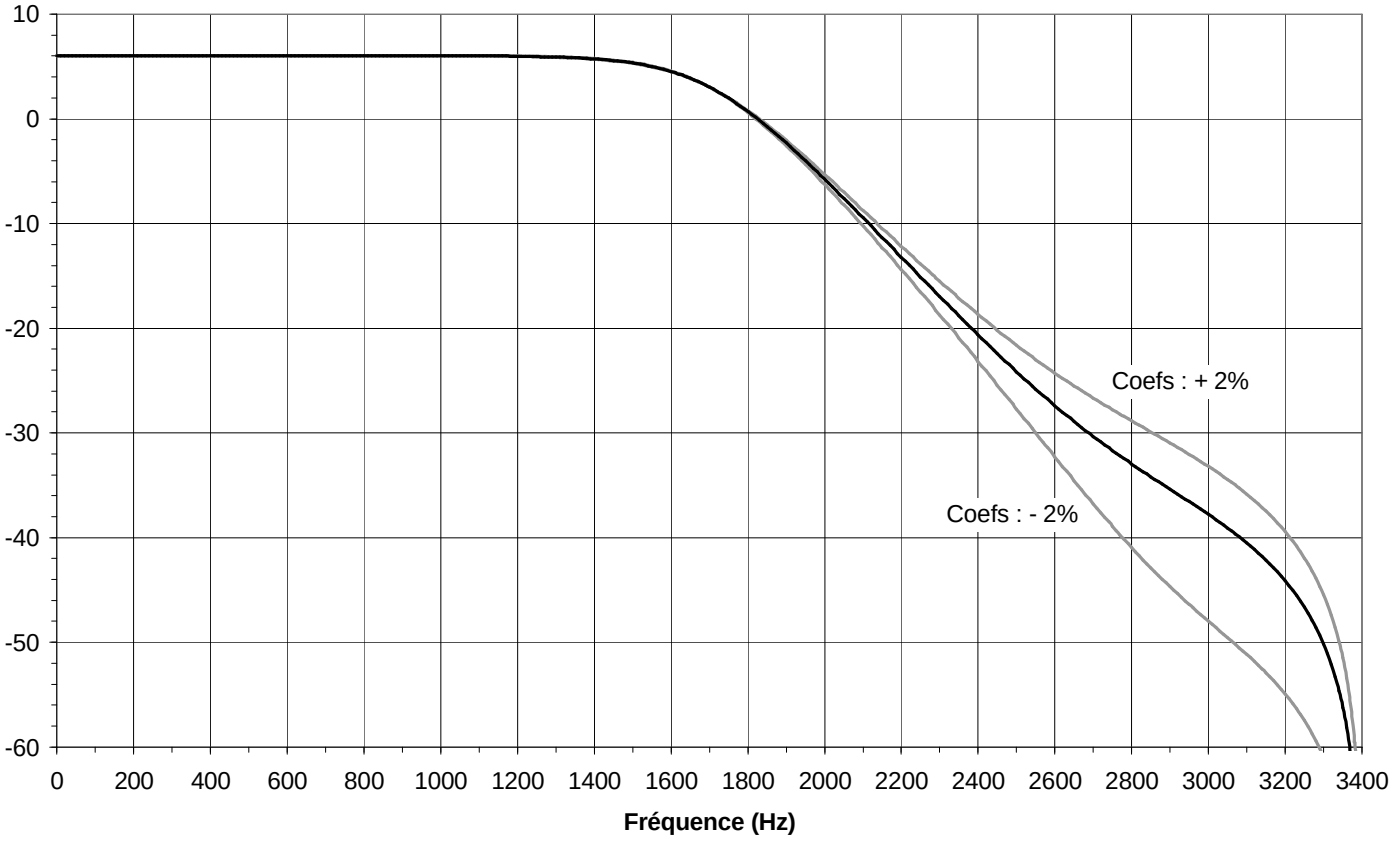
$$\frac{X(z)}{R9(z)} = \frac{K_1 \times Z^{-1}}{(K_1 - 1) \times Z^{-2} - 1} \qquad \frac{Y(z)}{R9(z)} = \frac{K_2 - 1}{K_2 \times Z^{-2} + 1} \qquad \frac{Y1(z)}{R9(z)} = -\frac{Z^{-2} + K_2}{K_2 \times Z^{-2} + 1}$$

$$\frac{X1(z)}{R9(z)} = \frac{Z^{-1} \times (K_1 - 1 - Z^{-2})}{Z^{-2} \times (K_1 - 1) - 1}$$

Et finalement : $\frac{R11(z)}{R9(z)} = X1 - Y1 = \frac{Z^{-1} \times (K_1 - 1 - Z^{-2})}{Z^{-2} \times (K_1 - 1) - 1} + \frac{K_2 + Z^{-2}}{K_2 \times Z^{-2} + 1}$

Simulation :

Cette fonction de transfert peut être simulée avec Excel en remplaçant Z par $e^{j \times \omega} = e^{j \times 2 \times \pi \times f}$



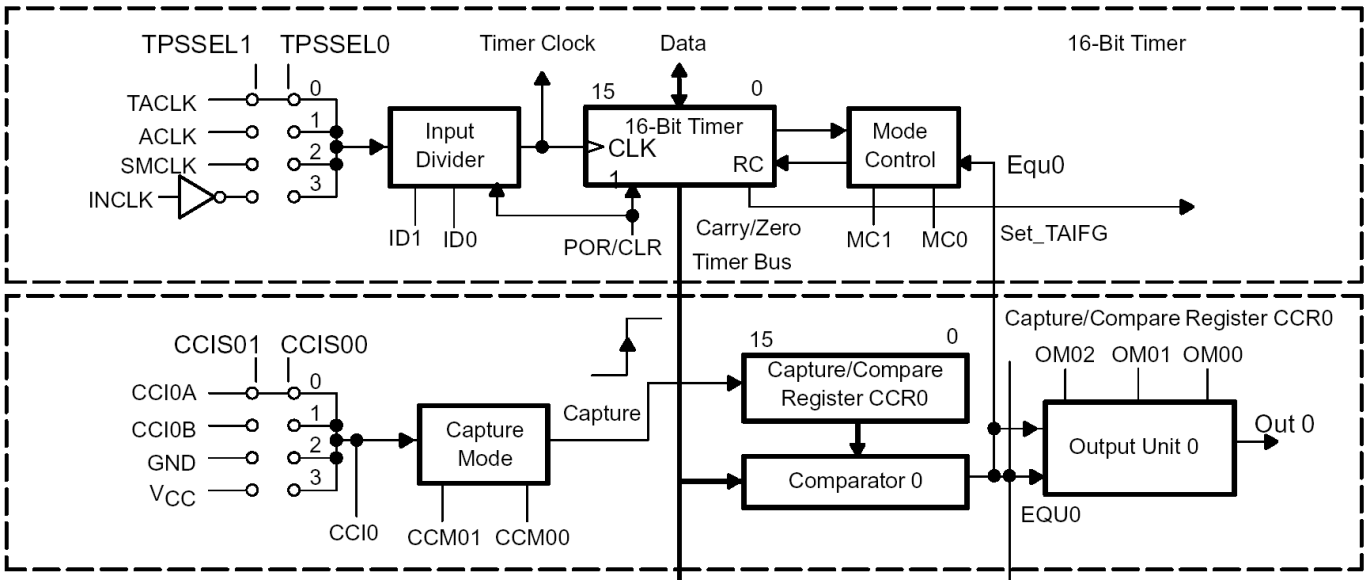
Les 2 courbes en grisés permettent d'évaluer l'influence d'un écart sur les coefficients K1 et K2. Une réduction de 2% par rapport à leurs valeurs nominales donne des meilleurs résultats, mais le programme nécessiterait alors des multiplications et ne serait plus traité suffisamment rapidement.

Note : Production de RXD à partir de R11 :
Le signal numérique R11 étant codé en complément à 2 sur 16 bits, le signal RXD est simplement égal à l'opposé de son bit de signe (comparaison au passage par zéro).
Pour une meilleure immunité au bruit, on peut réaliser un comparateur à hystérésis.

2. Réalisation de la fonction "démodulation FSK"

2.1 Production de la fréquence d'échantillonnage

La prise d'un échantillon, sa conversion et son traitement (fonctions F1 à F5) sont réalisés par un programme d'interruption. Celui-ci est activé régulièrement au double de la fréquence d'échantillonnage, soit 13600Hz.
Le "Timer A" du microcontroleur est configuré dans ce sens :

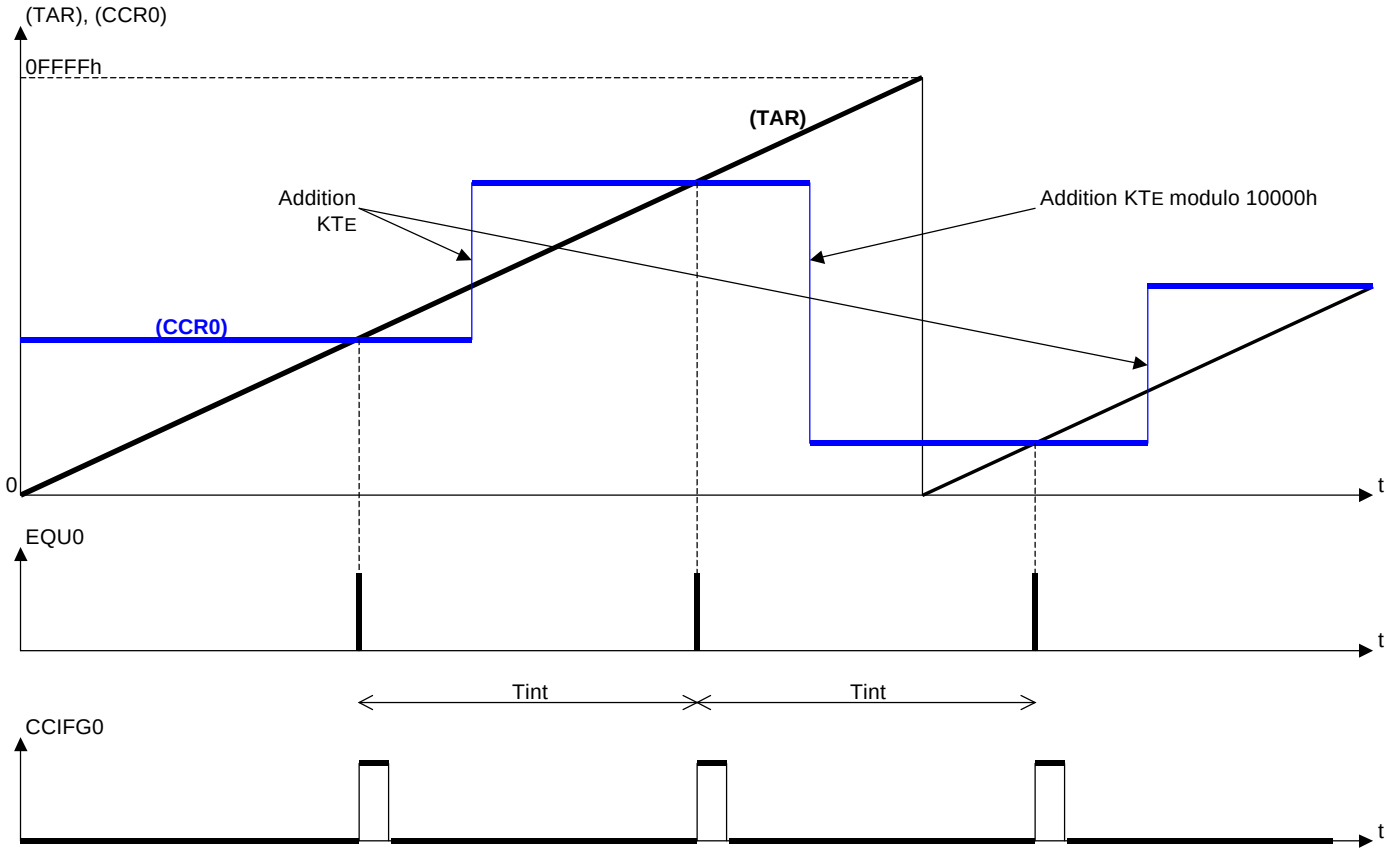


- Le multiplexeur de signaux d'horloge et le diviseur d'entrée sont configurés pour obtenir une fréquence d'horloge CLK du compteur TAR la plus élevée possible. Par exemple 4 MHz.
- Le compteur fonctionne en mode "continuous up"
- La fonction "Capture/Compare" est configurée en mode "Compare"
- EQU0 met à "1" l'indicateur d'interruption CCIFG0 du registre CCTL0
- L'indicateur CCIFG0 provoque une interruption.

Fonctionnement :

- Le compteur TAR s'incrémente à chaque période d'horloge modulo 10000h (16 bits)
- Le signal EQU0 passe à "1" quand l'état du compteur est égal au contenu du registre CCR0. L'impulsion sur EQU0 dure donc une période d'horloge CLK.
- L'activation de EQU0 provoque la mise à "1" de l'indicateur CCIFG et interrompt le programme en cours
- Le CPU du MSP430 exécute alors le programme d'interruption "TIM_A_Int"
- Avant de réaliser les fonctions F1 à F5 de la démodulation FSK, le programme d'interruption affecte d'abord le registre CCR0 pour que la prochaine interruption se produise après un délai d'une période d'échantillonnage T_E.

Chronogrammes typiques :



- (TAR) évolue normalement en marches d'escalier, mais l'échelle ne permet pas de les représenter
- EQU0 passe à "1" pendant une période d'horloge quand (TAR) = (CCR0)
- CCIFG0 est mis à "1" au même moment et provoque l'interruption du programme
- Le programme d'interruption "TIM_A_Int" est activé. Il remet automatiquement à "0" l'indicateur CCIFG0 et dure "un certain temps". Sa durée doit être inférieure à Tint.
- Au début de son déroulement, le programme "TIM_A_Int" lit le contenu du registre CCR0, lui ajoute la quantité KT_E sans se préoccuper de l'éventuel dépassement, et affecte (CCR0) avec le résultat. Pour atteindre le nouveau seuil (CCR0), le compteur doit augmenter de la valeur KT_E, donc compter autant de périodes d'horloge.
  On en déduit : **Tint = KT_E périodes d'horloge CLK**

  Exemple : avec une horloge de 4 MHz il faut obtenir : $Tint = \frac{T_E}{2} = \frac{1}{2 \times F_E} = \frac{1}{2 \times 6800} = \frac{KT_E}{4E^6}$

  Soit $KT_E = 294$

Éléments de programme associé :

Le programme proposé est adapté à un MSP430F1121 rythmé par une horloge de 4 MHz

Initialisations et boucle sans fin

Ne sont données ici que les lignes de programme associées à la fonction étudiée.

```
            bis.b #XTS,&BCSCTL1       ;Validation XT1 hautes fréquences
            bic   #OSCOFF,SR          ;Validation XT1 (par prudence)
Attend1     bic.b #OFIFG,&IFG1        ;Raz flag "Oscfault"
            mov   #0FFFh,R15          ;
Attend2     dec   R15                 ; > Délai de stabilisation de l'oscillateur
            jnz   Attend2             ;/
            bit.b #OFIFG,&IFG1        ;Test indicateur "Oscfault"
            jnz   Attend1
            mov.b #11001000b,&BCSCTL2 ;MCLK = SMCLK = XT1CLK
```

```
            mov   #0120h,&TACTL       ;Timer A : ACLK, continuous up. Pas d'int TAIE

            mov   #2010h,&CCTL0       ;Output Capture avec interruption
            mov   #294-1,&CCR0        ;Périodes int=4E6/294=13605Hz (env. 8x1700Hz)
*                                     ;On peut en fait affecter n'importe quelle val.
            . . .
            . . .
            eint                      ;Validation des interruptions au niveau CPU

BOUCLE      jmp   BOUCLE              ;Boucle sans fin. On attend l'interruption.
```

Début et fin du programme d'interruption "TIM_A_Int" :

```
TIM_A_Int   add #294-1,&CCR0          ;Périodes int=4E6/294=13605Hz (env 8x1700Hz)
            . . .
            . . .
            reti
```

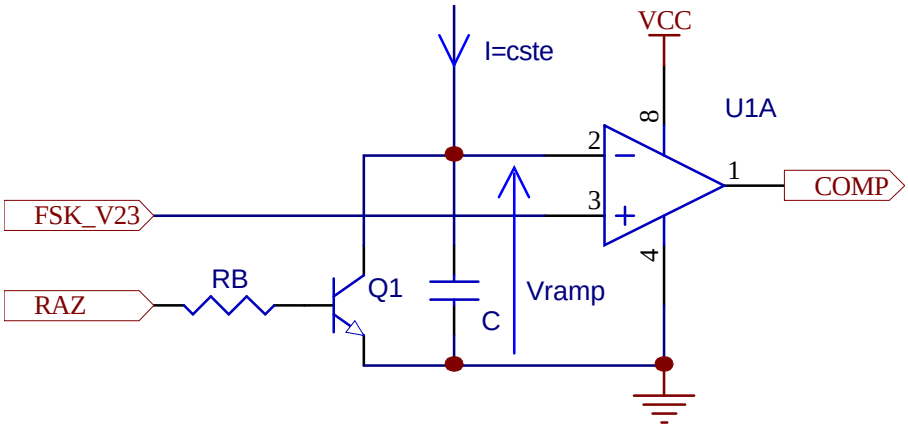
L'indicateur CCIFG0 est automatiquement mis à zéro à l'activation du programme d'interruption.

Vecteurs "Reset" et "Interruptions"

```
            org 0ffe0h    ;                                   Priorité
            dw  START     ;                                      0 (la plus faible)
            dw  START     ;                                      1
            dw  START     ;I/O Port P1                           2
            dw  START     ;I/O Port P2                           3
            dw  START     ;                                      4
            dw  START     ;                                      5
            dw  START     ;                                      6
            dw  START     ;                                      7
            dw  START     ;Timer A3, CCIFG1, CCIFG2, TAIFG       8
            dw  TIM_A_Int ;Timer A3, CCIFG0                      9
            dw  START     ;Watchdog timer                        10
            dw  START     ;Comparateur A                         11
            dw  START     ;                                      12
            dw  START     ;                                      13
            dw  START     ;NMI                                   14
            dw  START     ;POR, reset externe, watchdog          15 (la plus élévée)
            end
```

2.2 F1 : Conversion A → N

On utilise la technique du "simple rampe". La précision et la fidélité sont médiocres, mais largement suffisantes pour cette application.

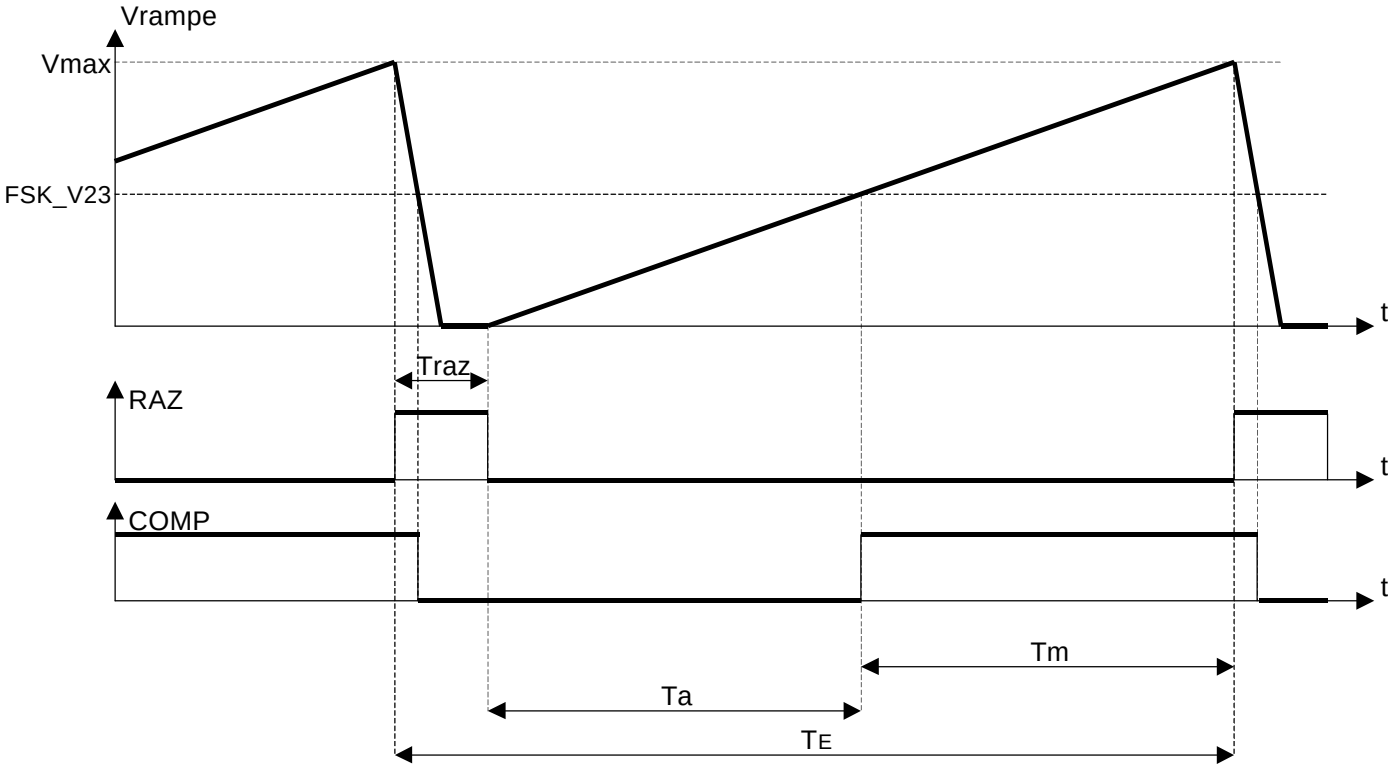


Le signal analogique à convertir est FSK_V23.
Le condensateur C est chargé à courant constant quand Q1 est bloqué. Il en résulte une rampe de pente :

$$k = \frac{dVramp}{dt} = \frac{I}{C}$$

C est régulièrement déchargé par Q1.

Chronogrammes typiques :



- La conversion A → N commence par l'étape A → T (conversion "tension" → "durée") :

  $$\boxed{Ta = k \bullet FSK_V23}$$

  Elle est réalisée par la structure présentée au début de ce §.
- La deuxième étape T → N (conversion "durée" → "nombre") est réalisée par des structures du "Timer A" du microcontroleur. En fait, celui-ci mesure la durée Tm.
  La durée Traz de la remise à zéro et la période T_E sont parfaitement connues et constantes. La connaissance de Tm permet donc de facilement calculer $\boxed{Ta = T_E - Tm - Traz}$.

Note : le MSP430 intègre un comparateur ("Comparateur A") qui peut être utilisé pour cette application ce qui réduit la quantité de composants externes.

Exploitation du "Timer A" :
Le § 2.1 décrit comment le "Timer A" est configuré pour produire des interruptions régulières au rythme de 13600 par seconde.

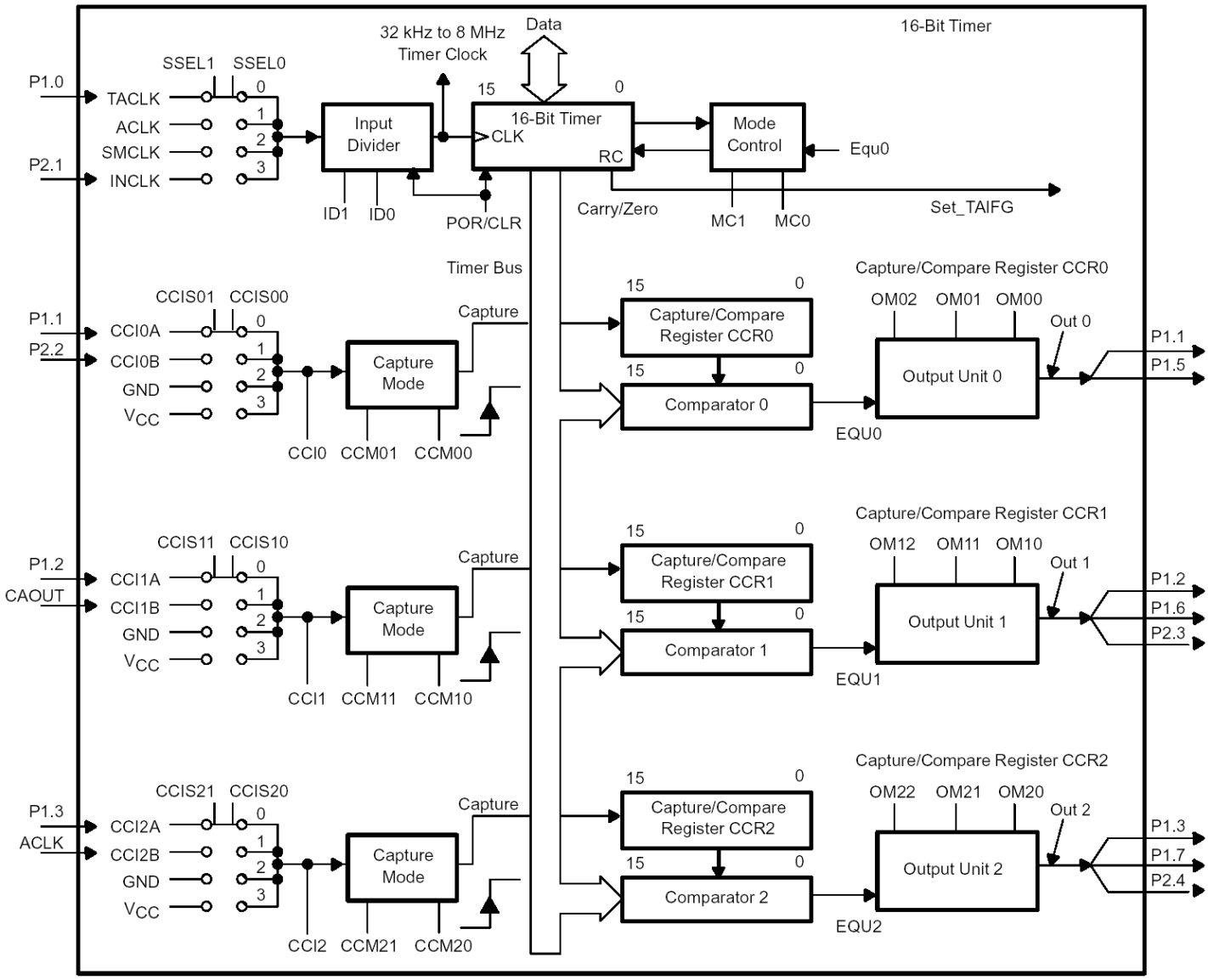
Le signal RAZ doit avoir une période $T_E = \frac{1}{F_E} = \frac{1}{6800}$. Il est donc produit toutes les 2 interruptions.

Le registre "CCR1" est utilisé en mode "Input Capture", activé par le flanc montant du signal COMP. Quand cet événement se produit, l'état correspondant du compteur TAR est mémorisé dans CCR1.

Conversion T → N :
La mesure de Tm est réalisée dans le programme d'interruption TIM_A_Int de la façon suivante :
- Le programme d'interruption est activé quand (TAR) atteint (CCR0) : voir § 2.1
- Au début du programme d'interruption, et une fois sur 2 :
  - une paire d'instructions créent l'impulsion RAZ
  - le microcontroleur affecte R7 avec (CCR1) – (CCR0)
    (CCR1) contient l'état du compteur TAR quand la rampe a atteint la tension FSK_V23 (↑ sur COMP). Cet événement s'est produit avant l'activation du programme d'interruption.
    (CCR0) contient l'état du compteur TAR à l'activation du programme d'interruption.
    La différence (CCR0) – (CCR1) représente donc la durée Tm en nombre de cycles d'horloge :

$R7 = (-Tm) \bullet F_{CLK}$ avec $F_{CLK} = 4MHz$ (par exemple)
$R7 = (Ta - T_E + Traz) \bullet F_{CLK}$
$R7 = Ta \bullet F_{CLK} + Cste = FSK_V23 \bullet k \bullet F_{CLK} + Cste$
Le calcul et la compensation de la constante "Cste" est inutile car le filtre passe-haut F2 qui suit élimine la composante continue de R7.

Timer A du MSP430F1121



Note : l'utilisation du registre CCR1 est dictée par le fait que la fonction "Input Capture" peut être activée par la sortie **CAOUT** du Comparateur A interne (voir schéma ci-dessus).

Éléments de programme associé :
Le programme proposé est adapté à un MSP430F1121 rythmé par une horloge de 4 MHz
Initialisations
Ne sont données ici que les lignes de programme associées à la fonction étudiée.

```
      mov   #5900h,&CCTL1          ;Input Capture flanc montant CAOUT sans interruption

      mov.b #00001000b,CACTL1      ;Comparateur A "on". Pas de référence interne
*                                  ;CA0 = "+" et CA1 = "-"
      mov.b #00001100b,CACTL2      ;Pas de filtre. P2.3 = CA0 , P2.4 = CA1
      mov.b #00011000b,CAPD        ;Pour limiter la consommation
      bis.b #00011000b,P2SEL       ;P2.3 = CA0 , P2.4 = CA1
      bis.b #RAZ,&P2DIR            ;RAZ est une const. qui indique la pos. du bit RAZ
```

Début et fin du programme d'interruption "TIM_A_Int" :

```
TIM_A_Int   add #294-1,&CCR0          ;Périodes int=4E6/294=13605Hz (env 8x1700Hz)
            bit #INTERRUPT_TOGGLE,R13 ;Test de l'indicateur INTERRUPT_TOGGLE
            jz  FILTRE_F4

DO_ADC      bic #INTERRUPT_TOGGLE,R13 ;A la prochaine int. : filtrage numérique
            bis.b #RAZ,&P2OUT         ;P2.1 = RAZ = "1" -> reset rampe
            bic.b #RAZ,&P2OUT         ;P2.1 = RAZ = "0" -> libération rampe

            mov &CCR1,R7
            sub &CCR0,R7              ;Mesure de la durée de la rampe dans R7
            . . .                     ;Suivent ici les fonctions F2, F3 et F4
            . . .
            reti
FILTRE_F4   bis #INTERRUPT_TOGGLE,R13
            . . .                     ;Suivent ici les fonctions F5 et "réception
            . . .                     ;série asynchrone".
            reti
```

Note : l'indicateur "INTERRUPT_TOGGLE" mémorisé dans le registre R13 permet d'activer la fonction F1 (conversion A $\rightarrow$ N) qu'une interruption sur 2 (pour respecter F_E = 6800 Hz).

Important : le résultat de la conversion (registre R7) doit être suffisamment faible pour être converti sur 8 bits en C2. Ceci est imposé par la fonction "multiplication" F4 qui ne peut traiter que des nombres codés sur 8 bits.

2.3 F2 : Filtre passe-haut

Cette fonction nécessite la mise en mémoire de l'échantillon précédent pour produire $R7(z) \bullet Z^{-1}$.
Le registre R10 du CPU est réservé pour cet usage.

Éléments de programme associé :

```
FONCTION_F2 mov R7,R6     ;R7 = R6 = échantillon actuel
            sub R10,R6    ;R10 = échantillon précédent : R6(z)=R7(z)-R7(z).Z-1
            mov R7,R10    ;Sauvegarde de l'échantillon actuel
```

2.4 F3 : Retard T_E

Cette fonction nécessite la mise en mémoire du contenu du registre R6 pour produire $R6_T$. La variable 16 bits "R6_T" est utilisée pour cet usage.

Éléments de programme associé :

```
FONCTION_F3 mov &R6_T,R7  ;R6_T dans R7 pour fonction F4 "multiplication"
            mov R6,&R6_T  ;Sauvegarde R6 actuel
```

2.5 F4 : Multiplication

Pour réduire le temps de calcul, le programme proposé réalise la multiplication des octets de poids faibles des registres R6 et R7. Le résultat sur 16 bits se retrouve dans le registre R9.
Toutes les variables sont codées en complément à 2.
Important : les valeurs de R6 et $R6_T$ aux sorties des fonctions F2 et F3 doivent être suffisamment faibles pour être codées sur 8 bits et éviter ainsi les dépassements dans la multiplication.

Éléments de programme associé :

```
************************************
* Multiplication 8bits x 8bits en C2 *
* R9 <- R6.b x R7.b                  *
************************************
* Registres modifiés : R7, R8, R9
* Durée : 75 cycles environ
MULTIPLI_F4     and   #0ffh,R6    ;Conversion 16 bits -> 8 bits en C2
                and   #0ffh,R7    ;Conversion 16 bits -> 8 bits en C2
                clr   R8
                tst.b R6
                jge   L$101
                swpb  R7
                sub   R7,R8
                swpb  R7
L$101           tst.b R7
                jge   MACU8
                swpb  R6
                sub   R6,R8
                swpb  R6
MACU8           mov   #1,R9
L$002           bit   R9,R6
                jz    L$01
                add   R7,R8
L$01            rla   R7
                rla.b R9
                jnc   L$002
                mov   R8,R9       ;Résultat de la multiplication dans R9
```

2.6 F5 : Filtre passe-bas

C'est la fonction la plus complexe du programme. Elle a été optimisée pour réduire le temps de calcul sans pour autant nuire aux performances.

Cette fonction nécessite la mise en mémoire FIFO des signaux R9, X et Y pour disposer de $R9(z)\bullet Z^{-1}$, $X(z)\bullet Z^{-2}$ et $Y(z)\bullet Z^{-2}$ (voir § 1.5). La variable WDF_PARAMS (5 mots de 16 bits) est utilisée pour cela :

- Adresse WDF_PARAMS : $R9(z)\bullet Z^{-1}$
- Adresse WDF_PARAMS + 2 : $Y(z)\bullet Z^{-1}$
- Adresse WDF_PARAMS + 4 : $Y(z)\bullet Z^{-2}$
- Adresse WDF_PARAMS + 6 : $X(z)\bullet Z^{-1}$
- Adresse WDF_PARAMS + 8 : $X(z)\bullet Z^{-2}$

Éléments de programme associé :

```
*********************************
* Fonction F4 : Filtre passe-bas *
*********************************
*  - Type Butterworth d'ordre 5
*  - Bande passante à -1dB : 1400Hz
*  - Ondulations dans la bande passante : < 1dB
*  - Bande coupée à -40dB : 2800Hz
* R9  = entrée (résultat de la fonction F4 multiplication)
* R11 = résultat du filtage
* Durée : 113 cycles
FILTRE_F4       bis #INTERRUPT_TOGGLE,R13
                mov #WDF_PARMS,R15
                mov 0(R15),R6      ;Echantillon précédent
                mov R9,0(R15)      ;Mémorisation de l'échantillon actuel
                mov 8(R15),R7      ; |
                sub R6,R7          ;/ R7 <- X(z-2) - E(z-1)
                mov R7,R8          ; |
                rra R8             ; |
                mov R8,R6          ; |
                rra R8             ; |
                rra R8             ;  > R6 <- [X(z-2) - E(z-1)]x(31/64)
```

```
rra R8              ; |
rra R8              ; |
rra R8              ; |
sub R8,R6           ;/
sub 8(R15),R6       ;R6 = X <- [X(z-2) - E(z-1)]x(31/64) - X(z-2)
                    ;          = -(31/64).E(z-1) - (33/64).X(z-2)
mov 6(R15),8(R15)   ;Mémorisation de X(z-1)
mov R6,6(R15)       ;Mémorisation de X
sub R7,R6           ;R6 <- X - [X(z-2) - E(z-1)]

mov 4(R15),R7       ; |
sub R9,R7           ;/ R7 <- Y(z-2) - E
mov R7,R8           ; |
rra R8              ; |
rra R8              ; |
rra R8              ; |
mov R8,R9           ;  > R9 <- [Y(z-2) - E]x(7/64)
rra R8              ; |
rra R8              ; |
rra R8              ; |
sub R8,R9           ;/
add 4(R15),R9       ;R9 <- [Y(z-2) - E]x(7/64) + Y(z-2)
                    ;      = -(7/64).E + (71/64).Y(z-2)
sub R9,R7           ;R7 = Y <- Y(z-2) - E + (7/64).E - (71/64).Y(z-2)
                    ;          = -(57/64).E - (7/64).Y(z-2)
mov 2(R15),4(R15)   ;Mémorisation de Y(z-1)
mov R7,2(R15)       ;Mémorisation de Y

sub R9,R6           ;Résultat final
mov R6,R11          ;R11 utilisé par la fonction "réception série
                    ;asynchrone"
```