

TP5 : Filtre RIF par DSP

Le but de ce TP est d'implanter le filtre RIF réjecteur (750 Hz) calculé dans le TP3 et de vérifier ses caractéristiques à l'aide d'un oscilloscope et d'un générateur de fonctions.

- 1) Créer un nouveau projet avec les fichiers *tp5.s* compléter le fichier source et simuler le avec les 5 coefficients simples de valeurs : 1 2 3 4 et 5 ($M=4$) .
 - Afficher les zones mémoires qui contiennent les échantillons, les coefficients ainsi que les registres utilisés dans le calcul (W0,W4,W5,W8,W10, ACCU,ACCH et ACCL)
 - Vérifier en pas à pas l'évolution du calcul et les valeurs des registres, ne passer à la suite que si cela fonctionne bien (vous pouvez forcer les valeurs des échantillons directement dans les zones mémoire pour vérifier les calculs) (**faire valider**)
- 2) Si les buffers circulaires et les calculs se font correctement, faire les modifications pour obtenir un filtre réjecteur 750Hz
 - copier les coefficients du filtre calculés avec matlab, etc ...
 - déterminer en simulation avec « trace » la durée de la routine d'interruption et l'intervalle de temps entre deux appels successifs (en cycles) pour vérifier si le calcul se fait suffisamment rapidement (le temps de calcul doit être inférieur à Tech)
 - Charger le programme dans la cible et vérifier les performances du filtre à l'aide de l'oscilloscope et du générateur de fonctions. (**à faire valider**)

;tp5.s

```

;-----
;La conversion de l'entrée analogique AN0 est déclenchée automatiquement par le TIMER3
;Lorsque la conversion est terminée le convertisseur déclenche une demande d'interruption
;la routine d'interruption calcule et envoie les 8bits de poids forts du résultat sur le port C
;Le programme principal après l'appel des sous programmes d'initialisation exécute une boucle infinie
;-----
;
;      .equ _33FJ16MC304, 1
;      .include "p33FJ16MC304.inc" ; des equ (defines) en quantité..... rend les programme plus lisibles si besoin
;-----

```

;Bits de configuration-----

; Watchdog off

```

      .section _FWDT.sec,code
      .global _FWDT
_FWDT: .pword 0x005f

```

; Oscillateur FRC+pll par défaut fcy=23Mhz (Vérifier à l'oscilloscope sur la broche CLK0 la fréquence cycle!!!!)

```

      .section _FOSCSEL.sec,code
      .global _FOSCSEL
_FOSCSEL: .pword 0x0081

```

;Les "defines" (equ) et les symboles globaux -----

```

      .equ          ECH,#5          ; nombre d'échantillons à traiter ( à modifier après la simulation )
      .global       _main           ; le startup se termine par goto _main
      .global       __ADC1Interrupt ; adresse du vecteur d'interruption ADC (voir .gld)

```

*; Réserve de 2*ECH octets en mémoire X . **Buffer_X** est l'adresse du premier élément-----*

```

      .section .xbss,bss,xmemory
      .align(128)
buffer_X: .space 2*ECH           ;taille en bytes

```

;On demande au "linker" de mettre les coefficients du filtre dans la RAM Y-----
;C'est le Startup (crt0_standart.s) qui va initialiser la RAM Y au démarrage

```

      .section .ydata,data,ymemory
      .align(128)
coeff: .word 0x0001, 0x000, 0x0003 ;5 coefficients pour la simulation ( à modifier après la simulation ... )
      .word 0x0004, 0x0005

```

 ; Le Programme principal

```

      .section .text          ;indique au linker d'utiliser la mémoire programme pour la suite
_main:
      rcall _init_dsp        ;sous-programmes d'initialisation
      rcall _init_timer3
      rcall _init_portC
      call _init_adc

```

INSSET DE ST QUENTIN

```
        rcall _init_int
boucle:      ; Le programme principal fonctionne en boucle infinie.
        nop      ; ne fait rien mais prend 1 cycle (no operation)
        bra boucle      ;

;fin du_programme principal.*****

;*****
;*****
; Routine de traitement de l'interruption ADC
;*****
__ADC1Interrupt:
        ;PARTIE à COMPLETER
                ;stocker l'entrée analogique dans le buffer_X
                ;calculer la sortie du filtre
                ;envoyer les poids forts de la valeur calculée sur le PORTD muni du réseau R/2R

        bclr IFS0,#13      ;réinitialise l'indicateur d'interruption ADC
        RETFIE

;*****
; Les sous-programmes d'initialisation
;*****
_init_dsp:
        mov #????,W0      ; Choisir W10 et W8 pour pointer sur les buffers circulaires ( ??? à remplacer)
        mov W0,MODCON
        nop
        mov #coeff,W0      ; adresse de début du buffer circulaire en RAM Y
        mov W0,YMODSRT
        ADD #(ECH*2)-1,W0      ; calcul de l'adresse de fin du buffer Y
        mov W0,YMODEND

        mov #????,X,W0      ; configurer le buffer circulaire en RAM X ( ??? à remplacer)
        mov W0,XMODSRT
        ADD #(ECH*2)-1,W0
        mov W0,XMODEND
        nop
        mov #0x????,W0      ;( ??? valeur à choisir)
        mov W0,CORCON
        mov #coeff,W10      ; W10 pointe sur les coefficients
        mov #buffer_X,W8      ; W8 pointe sur les échantillons
        return

_init_portC:
        ??? ????      ; Configurer les 8 bits de poids faible en sorties
        return

_init_timer3:
        mov #0x8000,W0
        mov W0,T3CON
        mov #???,W0      ;à calculer pour obtenir une fréquence d'échantillonnage de 11025Hz avec fcy=23MHz
        mov W0,PR3
        return

_init_adc:
        mov #0xfffe,W0
        and AD1PCFGL      ; mise à 0 du bit 0 par un ET avec W0 (fffe)
        mov #0x???,W0      ; à choisir
        mov W0,AD1CON3
        mov #0x0000,W0
        mov W0,AD1CON2
        mov W0,AD1CHS0
        mov #0x???,W0      ;utiliser de préférence le mode fractional signed
        mov W0,AD1CON1
        return

_init_int:
        bclr IFS0,#13
        bset IEC0,#13
        return

.end      ; fin du fichier
```

```
_init_dsp:

        mov #coeff,W0
        mov W0,YMODSRT      ;
        ADD #(ECH*2)-1,W0      ; calcul de l'adresse de fin
        mov W0,YMODEND

        mov #buffer_X,W0
        mov W0,XMODSRT      ;adresse de début du buffer X
        ADD #(ECH*2)-1,W0      ; calcul de l'adresse de fin
        mov W0,XMODEND

        mov #???,W0
        mov W0,CORCON      ; Autoriser la saturation et le mode virgule fixe
```

INSSET DE ST QUENTIN

```

        mov #coeff,W10          ; Initialisation des pointeurs W10 et W8
        mov #buffer_X,W8
        return

_init_portD:
        clr TRISD                ; portD en sortie
        return

_init_timer3:
        mov #0x8000,W0
        mov W0,T3CON
        mov #???,W0
        mov W0,PR3                ; Choisir la période du timer tel que fe = 11025 Hz
        return

_init_adc:
        clr ADPCFG
        mov #0x0400,W0
        mov W0,ADCON2
        mov #0x010A,W0
        mov W0,ADCON3
        mov #0x0000,W0
        mov W0,ADCHS
        mov #0x0001,W0
        mov W0,ADCSSL
        mov #0x8246,W0
        mov W0,ADCON1
        return

_init_int:
        bclr IFS0,#11
        bset IEC0,#11            ; Autorisation des interruptions ADC
        return
        .end                      ; fin du fichier
```