ΕΡΓΑΣΤΗΡΙΟ ΨΗΦΙΑΚΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΣΗΜΑΤΩΝ

Εργαστήριο 4

Ομάδα 09 – Group 1

ΕΠΩΝΥΜΟ	ΟΝΟΜΑ	ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ
ΔΑΣΟΥΛΑΣ	ΙΩΑΝΝΗΣ	1053711
ΔΟΥΡΔΟΥΝΑΣ	ΑΡΙΣΤΕΙΔΗΣ ΑΝΑΡΓΥΡΟΣ	1047398

A/D, D/A , Δειγματοληψία και FIR φίλτρα

Στόχος:

Η εξοικείωση με την ρύθμιση των περιφερειακών του C6713 DSK board για τη μετατροπή A/D και D/A

Σειριακή Θύρα

Ο AIC23 codec συνδέεται με την σειριακή θύρα McBSP1 του C6713 DSP. Επομένως, πρέπει να ρυθμίσετε την McBSP1 για να είστε σε θέση να χρησιμοποιήσετε τον codec. Μετά από αρχικοποίηση (reset) ο AIC23 κάνει συνεχώς A/D και D/A μετατροπή. Η ρύθμιση του AIC23 αλλάζει στέλνοντας κατάλληλα σήματα ελέγχου μέσω της McBSP0. Ο codec για αυτό το εργαστήριο έχει ρυθμιστεί για 16bit A/D και D/A stereo και δειγματοληψία στα 48KHz. Τα αποτελέσματα της δειγματοληψίας A/D καθώς και τα στοιχεία μετατροπής D/A οδηγούνται συνεχώς στη σειριακή θύρα McBSP1. Για να ρυθμίσετε την McBSP1 του DSP πρέπει να καταχωρήσετε τις κατάλληλες τιμές στους καταχωρητές που ελέγχουν τη σειριακή θύρα. Αυτοί είναι καταχωρητές μνήμης και οι φυσικές διευθύνσεις τους είναι:

Register Name	Address	Description
McBSP1_DRR	0x1900000	Data Receive Register
McBSP1_DXR	0x1900004	Data Transmit Register
McBSP1_SPCR	0x1900008	Serial Port Control Register
McBSP1_RCR	0x190000C	Receive Control Register
McBSP1_XCR	0x1900010	Transmit Control Register
McBSP1_SRGR	0x1900014	Sample Rate Generator Register
McBSP1_MCR	0x1900018	Multichannel Control Register
McBSP1_RCER	0x190001C	Receive Channel Enable Register
McBSP1_XCER	0x1900020	Transmit Channel Enable Register
McBSP1_PCR	0x1900024	Pin Control Register

Η συμπεριφορά της McBSP1 διαμορφώνεται με τη ρύθμιση των καταχωρητών SPCR, SRGR, PCR, RCR και XCR. Για αυτήν την ρύθμιση, οι ακόλουθες τιμές καταχωρητών είναι εκείνες που χρειάζονται να τεθούν.

Ασκήσεις:

Άσκηση 4.1 - Ρύθμιση McBSP1

Να δημιουργηθεί ένα CCS Project με τις κατάλληλες ρυθμίσεις και τα απαραίτητα αρχεία και να αναπτυχθεί μια ρουτίνα σε assembly ώστε να ρυθμιστεί η McBSP1.

Δημιουργήθηκε νέο CCS Project με τις απαιτούμενες ρυθμίσεις και συμπεριλήφθηκαν σε αυτό τα απαραίτητα αρχεία. Έπειτα, αναπτύχθηκε ρουτίνα σε assembly για την ρύθμιση του McBSP1. Πρόγραμμα:

```
RCR         .set 0x190000C  ; Address of RCR
XCR         .set 0x1900010  ; Address of XCR
a           .set 0x000000A0

            .def _entry
            .text


_entry:     MVKL  a, A1
            MVKH  a, A1
            MVKL  RCR, A2
            MVKH  RCR, A2
            STW   A1,*A2

            MVKL  XCR, A4
            MVKH  XCR, A4
            STW   A1, *A4

            IDLE
            .end
```

Στην αρχή οι απαραίτητες τιμές γίνονται set και μεταφέρονται σε καταχωρητές. Ύστερα, η τιμή A0 αποθηκεύεται στην διεύθυνση του RCR και του XCR.

Άσκηση 4.2 - Μετάδοση δεδομένων

Υποθέτοντας ότι τα 32-bit δεδομένα που πρέπει να μεταδοθούν μέσω της McBSP1 βρίσκονται στον A0, να γραφεί μια ρουτίνα σε γλώσσα assembly που να ελέγχει τον SPCR και να διαβιβάζει έπειτα τα δεδομένα στην McBSP1. Εάν ο DXR δεν είναι κενός, το πρόγραμμά πρέπει να περιμένει έως ότου τελειώσει η McBSP1 τη μεταβίβαση των υπόλοιπων δεδομένων.

Για μετάδοση δεδομένων χρησιμοποιείται ο DXR καταχωρητής που περιέχει τα δεδομένα εξόδου της σειριακής θύρας. Για να είναι δυνατή η εγγραφή στον DXR πρέπει να έχει τελειώσει η προηγούμενη διαβίβαση δεδομένων. Η κατάσταση του DXR φαίνεται από το bit νούμερο 17 του SPCR (XRDY bit). Όταν αυτό γίνεται 1, ο DXR μπορεί να πάρει νέα δεδομένα.

Figure 54. Serial Port Control Register (SPCR)

31						26	25	24
Reserved							FREE(1)	SOFT(1)
R-0							R/W-0	R/W-0
23	22	21		20	19	18	17	16
FRST	GRST	XINTM			XSYNCERR	XEMPTY	XRDY	XRST
R/W-0	R/W-0	R/W-0			R/W-0	R-0	R-0	R/W-0
15	14	13	12	11	10			8
DLB	RJUST		CLKSTP		Reserved			
R/W-0	R/W-0		R/W-0		R-0			
7	6	5		4	3	2	1	0
DXENA(1)	Reserved	RINTM			RSYNCERR	RFULL	RRDY	RRST
R/W-0	R-0	R/W-0			R/W-0	R-0	R-0	R/W-0

LEGEND: R = Read only; R/W = Read/Write; -n = value after reset

(1) Available only on C621x/C671x DSP and C64x DSP.

Πρόγραμμα:

```
SPCR        .set 0x1900008 ; SPCR ADDRESS
DXR         .set 0x1900004 ; DXR ADDRESS

            .def _entry
            .text

_entry:     MVKL SPCR, A2
            MVKH SPCR, A2
            MVKL 0x20000, A3      ;BIT 17 = 1
            MVKH 0x20000, A3
            MVKL DXR, A1
            MVKH DXR, A1

loop:       LDW *A2,B1 ; LOAD SPCR VALUE
            NOP 4
            AND B1, A3, B1   ;CHECK 17th BIT

            XOR B1, A3, B1   ;EXIT LOOP
            [B1]  B loop
            NOP 5

            STW A0,*A1    ; STORE VALUE TO DXR

            IDLE
            .end
```

Αρχικά, γίνονται set οι διευθύνσεις και μεταφέρονται σε καταχωρητές καθώς και ορίζεται ένας καταχωρητής με ενεργοποιημένο το bit 17 (A3). Έπειτα, με τον βρόχο που ακολουθεί ελέγχεται αν ο DXR είναι έτοιμος για εγγραφή. Συγκεκριμένα, γίνεται φόρτωση η τιμή της διεύθυνσης του SPCR σε έναν καταχωρητή B1 ο οποίος στην συνέχεια συγκρίνεται με τον A3 μέσω της εντολής AND απομονώνοντας με αυτόν τον τρόπο το bit 17. Μετά, με την XOR εντολή ανάμεσα σε B1 και A3, αν το XRDY bit

είναι 1, το αποτέλεσμα της XOR είναι 0, οπότε τελειώνει ο βρόχος με τον έλεγχο που ακολουθεί και αποθηκεύονται τα δεδομένα του A0 στον DXR χωρίς πρόβλημα. Αντίθετα, αν το XRDY bit είναι 0, η XOR εντολή δίνει 1 και με τον έλεγχο που ακολουθεί, επαναλαμβάνεται ο βρόχος, συνεχίζεται η μεταφορά δεδομένων, μέχρις ότου η XOR κάποια στιγμή να δώσει αποτέλεσμα 0.

Άσκηση 4.3 - Λήψη δεδομένων

Να γραφεί μια ρουτίνα σε γλώσσα assembly που να ελέγχει τον SPCR και να διαβάζει έπειτα τα δεδομένα από τον DRR. Εάν δεν υπάρχει κανένα δεδομένο στον DRR, το πρόγραμμά πρέπει να περιμένει έως ότου η McBSP1 λάβει τα νέα δεδομένα στον DRR.

Ο καταχωρητής DRR περιέχει τα στοιχεία της θύρας McBSP1 που παραλαμβάνονται από το pin των δεδομένων εισόδου. Αφότου ο AIC23 codec τελειώσει την A/D μετατροπή, το αποτέλεσμα διαβιβάζεται στην McBSP1 και καταχωρείται στον DRR. Όταν τα δεδομένα που λαμβάνονται είναι διαθέσιμα στον DRR για ανάγνωση από τον DSP, το δυαδικό ψηφίο 1 (RRDY bit) του καταχωρητή SPCR γίνεται 1.

Figure 54. Serial Port Control Register (SPCR)

31	30	29	28	27	26	25	24
Reserved						FREE(1)	SOFT(1)
R-0						R/W-0	R/W-0
23	22	21	20	19	18	17	16
FRST	GRST	XINTM		XSYNCERR	XEMPTY	XRDY	XRST
R/W-0	R/W-0	R/W-0		R/W-0	R-0	R-0	R/W-0
15	14	13	12	11	10	9	8
DLB	RJUST		CLKSTP		Reserved		
R/W-0	R/W-0		R/W-0		R-0		
7	6	5	4	3	2	1	0
DXENA(1)	Reserved	RINTM		RSYNCERR	RFULL	RRDY	RRST
R/W-0	R-0	R/W-0		R/W-0	R-0	R-0	R/W-0

LEGEND: R = Read only; R/W = Read/Write; -n = value after reset

(1) Available only on C621x/C671x DSP and C64x DSP.

Πρόγραμμα:

```
SPCR        .set 0x1900008 ; SPCR ADDRESS
DRR         .set 0x1900000 ; DRR ADDRESS

            .def _entry
            .text

_entry:     MVKL SPCR, A1
            MVKH SPCR, A1
            MVKL DRR, A2
            MVKH DRR, A2


loop:       LDW *A1,B0
            NOP 4
```

```
AND B0, 2, B0 ;CHECK FIRST BIT

XOR B0, 2, B0 ;EXIT LOOP

[B0]  B loop
NOP 5

LDW *A2, A0 ;LOAD VALUE TO A0
NOP 4

IDLE
.end
```

Ανάλογα με την προηγούμενη άσκηση, αρχικά, γίνονται set οι διευθύνσεις και μεταφέρονται σε καταχωρητές. Έπειτα, με τον βρόχο που ακολουθεί ελέγχεται αν ο DPR είναι έτοιμος για εγγραφή. Συγκεκριμένα, γίνεται φόρτωση η τιμή της διεύθυνσης του SPCR σε έναν καταχωρητή B0 ο οποίος στην συνέχεια συγκρίνεται με τον αριθμό 2 (ενεργοποιημένο bit νούμερο 1) μέσω της εντολής AND απομονώνοντας με αυτόν τον τρόπο το bit 1. Μετά, με την XOR εντολή ανάμεσα σε B0 και 2, αν το RRDY bit είναι 1, το αποτέλεσμα της XOR είναι 0, οπότε τελειώνει ο βρόχος με τον έλεγχο που ακολουθεί και φορτώνονται τα δεδομένα στον A0 χωρίς πρόβλημα. Αντίθετα, αν το RRDY bit είναι 0, η XOR εντολή δίνει 1 και με τον έλεγχο που ακολουθεί, επαναλαμβάνεται ο βρόχος, συνεχίζεται η μεταφορά δεδομένων, μέχρις ότου η XOR κάποια στιγμή να δώσει αποτέλεσμα 0.

Άσκηση 4.4 - A/D και D/A βρόχος (polling)

Να γραφεί ένα πρόγραμμα σε γλώσσα assembly που να διαβάζει τα δεδομένα μετατροπής A/D και να γράφει άμεσα πίσω τα ίδια δεδομένα για τη μετατροπή D/A. Με τη σύνδεση μιας ηχητικής πηγής και ηχείων στο DSK board, πρέπει να ακουστεί ήχος εισόδου. Ανάλογα με το επίπεδο έντασης των σημάτων εισόδου, μπορεί να είναι απαραίτητο να πολλαπλασιαστεί η είσοδος με μια σταθερά, συνήθως από 4 έως 8, πριν από τη μετατροπή D/A για να ενισχυθεί το σήμα.

Πρόγραμμα:

```
SPCR        .set 0x1900008 ; SPCR ADDRESS
DRR         .set 0x1900000 ; DRR ADDRESS.
DXR         .set 0x1900004 ; DXR ADDRESS

            .def _entry
            .text

_entry:     MVKL DXR, A0
            MVKH DXR, A0

            MVKL DRR, A1
            MVKH DRR, A1
```

```
            MVKL SPCR, A2
            MVKH SPCR, A2

            MVKL 0x20000, A3
            MVKH 0x20000, A3

loop:

loopDRR:
            LDW *A2, B0
            NOP 4

            AND B0, 2, B0 ;CHECK FIRST BIT
            XOR B0, 2, B0 ;EXIT LOOP
            [B0]  B loopDRR
            NOP 5

            LDW *A1, A4 ;LOAD VALUE TO  A4
            NOP 4

loopDXR:
            LDW *A2, B1      ; LOAD SPCR VALUE
            NOP 4

            AND B1, A3, B1  ;CHECK 17th BIT
            XOR B1, A3, B1  ;EXIT LOOP
            [B1]  B loopDXR

            NOP 5
            STW A4,*A0   ; STORE VALUE TO DXR

            B loop
            NOP 5

            IDLE
            .end
```

Στην άσκηση αυτή, συνδυάζονται τα ερωτήματα 4.2, 4.3 σε έναν συνεχή βρόχο ώστε να γράφονται και να διαβάζονται συνεχώς δεδομένα. Αφού αποθηκευτούν οι απαραίτητες τιμές σε καταχωρητές, χρησιμοποιείται ο συνεχής βρόχος loop που εμπεριέχει τους δύο βρόχους όπως περιγράφονται στις παραπάνω ασκήσεις, δηλαδή τον βρόχο loopDRR για φόρτωση δεδομένων και τον βρόχο loopDXR για αποθήκευση δεδομένων (pooling). Κατά την διάρκεια του εργαστηρίου, με τη σύνδεση μιας ηχητικής πηγής και ηχείων στο DSK board, ακούστηκε επιτυχώς ο ήχος εισόδου.

Άσκηση 4.5 - A/D και D/A βρόχος (με διακοπές)

Να γραφεί ένα πρόγραμμα σε γλώσσα assembly που να διαβάζει τα δεδομένα μετατροπής A/D και να γράφει άμεσα πίσω τα ίδια δεδομένα για τη μετατροπή D/A. Χρησιμοποιώντας την RINT1 διακοπή, το αποτέλεσμα A/D διαβάζεται μέσω της

ρουτίνας εξυπηρέτησης διακοπής. Το πρόγραμμα απλά περιμένει την εμφάνιση αιτήσεων διακοπών. Επειδή και οι μετατροπές A/D και D/A εμφανίζονται σε υψηλό ρυθμό (48 kHz), αρκεί να εξυπηρετηθεί μόνο μια διακοπή (π.χ. RINT1) για να ληφθεί και στη συνέχεια να αποσταλεί το ψηφιακό σήμα.

Πρόγραμμα:

```
DRR         .set 0x1900000   ; DRR ADDRESS
DXR         .set 0x1900004   ; DXR ADDRESS
RCR         .set 0x190000C   ; RCR ADDRESS
XCR         .set 0x1900010   ; XCR ADDRESS

IML_pointer      .set 0x019c0004
IMH_pointer      .set 0x019c0000

            .def _entry
            .def Int_ISR
            .text

_entry:     MVKL 0xFFFF, B0 ; CLEAR ALL PREVIOUS INTERRUPTS
            MVKH 0xFFFF, B0
            MVC B0, ICR

            MVKL  DXR, A0  ; DXR POINTER
            MVKH  DXR, A0

            MVKL  DRR, A1    ;DRR POINTER
            MVKH  DRR, A1

            MVKL IML_pointer, A3 ; IML POINTER
            MVKH IML_pointer, A3

            MVKL IMH_pointer, A4 ; IMH POINTER
            MVKH IMH_pointer, A4

            MVKL  RCR, A5    ; RCR POINTER
            MVKH  RCR, A5

            MVKL  XCR, A6    ; XCR POINTER
            MVKH  XCR, A6

            MVKL  0xA0, A7   ; 0xA0
            MVKH  0xA0, A7

            STW   A7,*A5     ; SET RCR, XCR
            STW   A7,*A6

            MVKL 0X13, B1    ; SET NMI AND INT4 TO 1
            MVC B1, IER

            MVKL 1, B2       ;SET GIE TO 1
            MVC B2, CSR

            LDW *A3, A8      ; INT4 -> RINT1
```

```
            NOP 4
            SET A8, 0, 3, A8
            STW A8, *A3

loop:       B loop
            NOP 5

Int_ISR:    LDW *A1, A9
            NOP 4
            STW A9, *A0
            B IRP
            NOP 5

            .end
```

Αρχικά, καθαρίζονται όλα τα προηγούμενα interrupts, γίνονται set οι απαραίτητες διευθύνσεις και μεταφέρονται σε καταχωρητές, γίνονται 1 τα bit των INT4, NMI και GIE. Έπειτα, ορίζεται στο INT4 διακοπή RINT4. Ακολουθεί ένας ατέρμονας βρόχος για να παρατηρείται η συμπεριφορά του προγράμματος και κατά πόσο εκτελείται το interrupt, ενώ μετά είναι οι εντολές του interrupt (Int_ISR). Εκεί, γίνεται φόρτωση της τιμής του DRR σε έναν καταχωρητή και μετά αποθήκευση της τιμής αυτής στον DXR. Με την εντολή B IRP το πρόγραμμα επιστρέφει στην κανονική του ροή, δηλαδή σε αυτήν την περίπτωση, στον ατέρμονα βρόχο. Με αυτό τον τρόπο αποφεύγονται οι συνεχείς επαναλήψεις και έλεγχοι που έχει η μέθοδος του pooling.

Για να λειτουργήσει το παραπάνω πρόγραμμα, έγινε τροποποίηση του αρχείου vectors, ώστε το INT4 να οδηγεί με άλμα στο interrupt Int_ISR. Τροποποιημένο αρχείο vectors:

```
.title  "vectors.asm"


            .ref Int_ISR
            .ref  _c_int00

            .sect      "vectors"

rst:    mvkl    .s2   _c_int00,b0
        mvkh    .s2   _c_int00,b0
        b  .s2   b0
        nop
        nop
        nop
        nop
        nop


INT_1:
            nop
            nop
```

```
            nop
            nop
            nop
            nop
            nop
            nop

                        .
                        .
                        .
                        .

INT_4:
      B Int_ISR
      nop
      nop
      nop
      nop
      nop
      nop
      nop

                        .
                        .
                        .
                        .
                        .
                        .
                        .
                        .
```

Άσκηση 4.6 - Αποθηκεύοντας δεδομένα στον buffer

1.Να οριστούν δύο buffersμεγέθους 512 bytes ο καθένας, έναν για κάθε κανάλι με όνομα buffer_leftκαι buffer_right.

2.Κατόπιν, να εισαχθούν οι κατάλληλες γραμμές κώδικα έτσι ώστε κάθε φορά που ο DSP διαβάζει από τον codec, η τιμή των 32-bitδεδομένων (χωρισμένη σε 2 half-words αντίστοιχα για το αριστερό και το δεξιό κανάλι) να αποθηκεύεται κατάλληλα στους 2 buffers.

3.Να αποθηκευθούν στους buffers όλα τα δεδομένα εισόδου αρχίζοντας από την κορυφή. Όταν ο buffer είναι πλήρης, τα δεδομένα να γράφονται από την κορυφή, γράφοντας επάνω στα παλιά δεδομένα.

4.Να γραφεί ο κώδικα και τρέξει τον στο board. Να εισαχθεί ένα σήμα ημίτονου12 kHz στον codec χρησιμοποιώντας κάποιο λογισμικό tonegenerator, είτε από τον υπολογιστή του εργαστηρίου είτε από κάποιο smartphone.

5.Ενώ η είσοδος παραμένει συνδεδεμένη, να γίνει pause στον DSP. Οι τιμές δειγμάτων παραμένουν καταχωρημένες στις θέση μνήμης όπου δείχνουν τα buffer_leftκαι buffer_right.

Ο κώδικας που αναπτύχθηκε είναι ο ακόλουθος:

```
SPCR  .set 0x1900008 ; SPCR ADDRESS
DRR   .set 0x1900000 ; DRR ADDRESS
DXR   .set 0x1900004 ; DXR ADDRESS
RCR   .set 0x190000C  ; RCR ADDRESS
XCR         .set 0x1900010 ; XCR ADDRESS

IML_pointer      .set 0x019c0004
IMH_pointer      .set 0x019c0000

buffer_left           .space 512
buffer_right     .space 512

            .def _entry
            .def Int_ISR
            .def buff
            .text

_entry:
            MVKL 0xFFFF,B0 ; CLEAR ALL PREVIOUS INTERRUPTS
            MVKH 0xFFFF,B0
            MVC B0,ICR

            MVKL  DXR,A0  ; DXR POINTER
            MVKH  DXR,A0

            MVKL  DRR,A1 ;DRR POINTER
            MVKH  DRR,A1

            MVKL IML_pointer, A3 ; IML POINTER
            MVKH IML_pointer, A3

            MVKL IMH_pointer, A4 ; IMH POINTER
            MVKH IMH_pointer, A4

            MVKL  RCR,A5     ; RCR POINTER
            MVKH  RCR,A5

            MVKL  XCR,A6     ; XCR POINTER
            MVKH  XCR,A6

            MVKL  0xA0,A7    ; 0xA0
            MVKH  0xA0,A7

            STW         A7,*A5  ;SET RCR, XCR
            STW     A7,*A6
```

```
            MVKL 0X13,B1 ;SET  NMI AND INT4 TO 1
            MVC B1,IER

            MVKL 1,B2 ;SET GIE TO 1
            MVC B2,CSR

            MVKL buffer_left, A8 ;STORE LEFT BUFFER
            MVKH buffer_left, A8

            MVKL buffer_right, A9 ;STORE RIGHT BUFFER
            MVKH buffer_right, A9

            MVK 0x100, B0 ;COUNTER

            LDW *A3 ,A10
            NOP 4
            SET A10,0,3,A10
            STW A10, *A3

loop: B loop
            NOP 5

Int_ISR:
            [!B0] B buff
            NOP 5
            LDW *A1, A10
            NOP 4
            STW A10, *A0
            MV A10, A11
            SHR A11, 16, A11
            STH A10, *A9++
            STH A11, *A8++
            SUB B0, 1, B0
            B IRP
            NOP 5

buff:
            MVKL buffer_left, A8 ;STORE LEFT BUFFER
            MVKH buffer_left, A8

            MVKL buffer_right, A9 ;STORE RIGHT BUFFER
            MVKH buffer_right, A9

            MVK 0x100, B0 ;COUNTER

            B Int_ISR
            NOP 5

            .end
```

Αρχικά , γίνονται οι απαραίτητες αρχικοποιήσεις που αφορούν το codec καθώς , την ενεργοποίηση του interrupt και την δέσμευση του buffer. Όπως ζητείται από την

εκφώνηση δημιουργούνται δύο buffer , ένας για το αριστερό κανάλι και ένας για το δεξί μέσω τον εντολών buffer_left .space 512 και buffer_right.space 512. Αφού γίνει η δέσμευση του χώρου στη μνήμη αποθηκεύονται οι διευθύνσεις των δύο buffer στους registers A8 και A9 αντίστοιχα. Στη συνέχεια ορίζεται ένας counter 256 επαναλήψεων στον register B0 ενώ ρυθμίζεται και το αντίστοιχο interrupt στο INT 4 μέσω των :

```
LDW *A3 ,A10
NOP 4
SET A10,0,3,A10
STW A10, *A3
```

Αφού γίνουν οι απαραίτητες ρυθμίσεις, τοποθετείται ένας ατέρμονας βρόχος έτσι ώστε η CPU να μην μπαίνει σε κατάσταση IDLE και να μπορεί να εκτελεστεί η διακοπή. Έπειτα , ορίζεται ο κώδικας που θα εκτελεστεί στην διακοπή:

```
Int_ISR:
            [!B0] B buff
            NOP 5
            LDW *A1, A10
            NOP 4
            STW A10, *A0
            MV A10, A11
            SHR A11, 16, A11
            STH A10, *A9++
            STH A11, *A8++
            SUB B0, 1, B0
            B IRP
            NOP 5

buff:
            MVKL buffer_left, A8 ;STORE LEFT BUFFER
            MVKH buffer_left, A8

            MVKL buffer_right, A9 ;STORE RIGHT BUFFER
            MVKH buffer_right, A9

            MVK 0x100, B0 ;COUNTER

            B Int_ISR
            NOP 5

            .end
```

Ο κώδικας της διακοπής ξεκινάει με ένα branch στο Label buff. Η λειτουργεία που επιτελεί το κομμάτι κώδικα που βρίσκεται μέσα στο buff αφορά την δημιουργία ενός

buffer , την αποθήκευση της διεύθυνσης στους δύο registers A8 , A9 και την αρχικοποίηση του μετρητή B0 και το Jump πίσω στο interrupt.

Το branch υπό συνθήκη που γίνεται στο interrupt ([!B0] B buff) ουσιαστικά ελέγχει ότι το περιεχόμενο του counter B0 είναι διάφορο του μηδενός , καθώς εάν ο counter λάβει την τιμή μηδέν σημαίνει ότι ο buffer που έχει δημιουργηθεί είναι γεμάτος. Αφού γίνει ο παραπάνω έλεγχος , φορτώνονται τα δεδομένα του DRR στους register A10 και A11 και στην συνέχεια τα δεδομένα χωρίζονται με την χρήση της εντολής SHR A11, 16, A11 η οποία μετακινεί κατά 16 θέσεις το περιεχόμενο του A11 και στο αποθηκεύει στον A11. Έτσι υπάρχουν δύο registers και ο ένας περιέχει τα 16 MSB του περιεχομένου του DRR ενώ ο άλλος τα 16 LSB του ΔΡΡ. Στη συνέχεια τα δεδομένα αποθηκεύονται στις διευθύνσεις των πρώτων στοιχείων του κάθε buffer και οι pointers ου δείχνουν δε αυτές τις διευθύνσεις αυξάνουν την τιμή τους κατά ένα , αφού πρώτα γίνει η αποθήκευση. Τέλος η τιμή του B0 ελαττώνεται κατά 1 και γίνεται branch στον καταχωρητή IRP.

Άσκηση 4.7 – Υλοποίηση FIR:

1. Γράψτε μια ρουτίνα assembly που υλοποιεί το FIR φίλτρο τροποποιώντας το πρόγραμμα εσωτερικού γινομένου που έχετε γράψει στο εργαστήριο 2.

2. Συνδυάστε τη λειτουργία φιλτραρίσματος FIR με την είσοδο δεδομένων από τον codec με την μέθοδο των διακοπών που γράψατε στο εργαστήριο

3. Είναι απαραίτητο να αρχικοποιήσετε τον AIC23 codec όπως κάνατε στα πρώτα ερωτήματα του εργαστηρίου. Ο κώδικάς σας θα πρέπει να φιλτράρει τα δεδομένα που έρχονται από τον codec και να στέλνει στην έξοδο κάθε φορά το αποτέλεσμα (real-time λειτουργία). Και τα δύο κανάλια υποβάλλονται σε επεξεργασία. Χρησιμοποιήστε τα αρχεία συντελεστών που σας παρέχονται. Αρχικά εφαρμόστε το lowpass φίλτρο τάξης 40 με αποκοπή στα 10kHz που έχει δημιουργηθεί με τη firpm. 3. Συνδέστε την έξοδο του υπολογιστή ή κάποιου smartphone που διαθέτουν λογισμικό tone generator, με την είσοδο (line in) του codec. Δημιουργήστε ένα ημιτονοειδές σήμα εισόδου στα 5kHz. Σχεδιάστε (γραφικά) μέσω του CCS και παρατηρήσετε την είσοδο και την έξοδο του codec.

4. Εισάγετε λευκό θόρυβο (wn.wav στο eclass) στην είσοδο του codec και παρατηρήστε το φάσμα της εξόδου. Μπορείτε να δείτε την απόκριση συχνότητας του φίλτρου;

5. Επαναλάβετε τα παραπάνω βήματα χρησιμοποιώντας όλα τα άλλα φίλτρα που σας παρέχονται.

Πρόγραμμα:

```
            .align 512
SPCR  .set 0x1900008 ; SPCR ADDRESS
```

```
DRR   .set 0x1900000 ; DRR ADDRESS
DXR   .set 0x1900004 ; DXR ADDRESS
RCR   .set 0x190000C ; RCR ADDRESS
XCR         .set 0x1900010 ; XCR ADDRESS

IML_pointer      .set 0x019c0004
IMH_pointer      .set 0x019c0000

buffer_left      .space 512
buffer_right     .space 512

            .def _entry
            .def int
            .ref coef
            .text

_entry:
            MVKL 0xFFFF,B0 ; CLEAR ALL PREVIOUS INTERRUPTS
            MVKH 0xFFFF,B0
            MVC B0,ICR
            ZERO B0

            MVKL  DXR,A0  ; DXR POINTER
            MVKH  DXR,A0

            MVKL  DRR,A1 ;DRR POINTER
            MVKH  DRR,A1

            MVKL IML_pointer, A3 ; IML POINTER
            MVKH IML_pointer, A3

            LDW *A3,A5  ; RINT1 -> INT4
            NOP 4
            SET A5,0,3,A5
            STW A5,*A3
            ZERO A5

            MVKL  RCR,A5     ; RCR POINTER
            MVKH  RCR,A5

            MVKL  XCR,A6     ; XCR POINTER
            MVKH  XCR,A6

            MVKL  0xA0,A7    ; 0xA0
            MVKH  0xA0,A7

            STW        A7,*A5  ;SET RCR, XCR
            STW     A7,*A6

            ZERO A5
            ZERO A6
            ZERO A7

            MVKL 0X13,B1 ;SET  NMI AND INT4 TO 1
            MVC B1,IER
```

```
            MVKL 1,B2 ;SET GIE TO 1
            MVC B2,CSR

            ZERO B1
            ZERO B2

            MVKL buffer_left, A8 ;STORE LEFT BUFFER
            MVKH buffer_left, A8
            MV A8, B4

            MVKL buffer_right, A9 ;STORE RIGHT BUFFER
            MVKH buffer_right, A9

            MVK 0x100, B2 ;COUNTER = 100

            MVKL coef,A13 ; COEFF ADRESS
            MVKH coef,A13

loop:
            B loop
            NOP 5

int:
            [!B2] B re_buff
            NOP 5

            LDW *A1,A4
            NOP 4

            ZERO A14
            ZERO A15

            STH A4,*A8++     ; LEFT BUFFER STORE (INPUT)
            MV A8,A7         ; SAVE CURRENT BUFFER POSITION

            MVKL 0x200,A11
            MVKH 0x200,A11

            MVKL 0x28,B0
            MVKH 0x28,B0

            SUB B2,1,B2           ;BUFFER HALFWORD COUNTER


check:
            SUB  A7,B4,B1    ;FIND IF BUFFER POSITION IS VALID
            ADD B4,A11,A12   ;BUFFER END
            [!B1] B new_addr
            NOP 5

conv:
            LDH *--A7,A4
            LDH *+A13[A14],A5
            NOP 4
            MPY A4,A5,A6
            NOP
```

```
            SHR A6,15,A6     ;Q-15 FORMAT
            ADD A6,A15,A15   ;ADD RESULT

            ADD A14,1,A14    ;COEFF ELEMENT COUNTER
            SUB B0,1,B0           ;COEFF COUNTER
            [B0]  B check
            NOP 5
            STH A15,*A9++    ;STORE RESULT IN RIGHT BUFFER, THEN
INCREMENT

            B IRP
            NOP 5

re_buff:
            MVKL buffer_left,A8
            MVKH buffer_left,A8
            MVKL buffer_right,A9
            MVKH buffer_right,A9
            MVK 0x100,B2
            B int
            NOP 5

new_addr:
            MV A12,A7
            B conv
            NOP 5
```

Στο πρώτο κομμάτι του προγράμματος γίνονται τα απαραίτητα set και οι αρχικοποιήσεις που αφορούν το interrupt και τις μεταβλητές που θα χρησιμοποιηθούν. Ακολουθεί το interrupt “int”:

```
 int:
            [!B2] B re_buff
            NOP 5

            LDW *A1,A4
            NOP 4

            ZERO A14
            ZERO A15

            STH A4,*A8++     ; LEFT BUFFER STORE (INPUT)
            MV A8,A7         ; SAVE CURRENT BUFFER POSITION

            MVKL 0x200,A11
            MVKH 0x200,A11

            MVKL 0x28,B0
            MVKH 0x28,B0

            SUB B2,1,B2           ;BUFFER HALFWORD COUNTER
```

Το πρόγραμμα μπαίνει στο interrupt όταν λαμβάνει κάποια είσοδο, η οποία έπειτα φορτώνεται στον καταχωρητή A4 μέσω του DRR και αποθηκεύεται στον αριστερό buffer, ο οποίος είναι ο buffer εισόδου. Επίσης με την εντολή MV, σώζεται κάθε φορά η τρέχουσα θέση του buffer ενώ ορίζονται και δύο μετρητές κάθε φορά, ο A11 που είναι ίσος με το μέγεθος των buffers και ο B0 που είναι ίσος με το μέγεθος του φίλτρου (40 στοιχεία, εκτός από το bandpass που έχει 80, όπου αλλάζει το συγκεκριμένο σημείο στον κώδικα). Τέλος, κάθε φορά αφαιρείται ο αριθμός 1 από τον B2, ο οποίος αρχικοποιείται ως 256 = 512/2, δηλαδή είναι ίσος με τον μέγιστο αριθμό halfwords που μπορούν να χωρέσουν στους buffers. Όταν αυτός μηδενιστεί, γίνεται branch στο κομμάτι re_buff όπου επαναρχικοποιούνται οι buffers:

```
re_buff:
           MVKL buffer_left,A8
           MVKH buffer_left,A8
           MVKL buffer_right,A9
           MVKH buffer_right,A9
           MVK 0x100,B2
           B int
           NOP 5
```

Μετά, γίνεται η συνέλιξη:

```
check:
           SUB  A7,B4,B1   ;FIND IF BUFFER POSITION IS VALID
           ADD B4,A11,A12  ;BUFFER END
           [!B1] B new_addr
           NOP 5
```

Αρχικά, στο κομμάτι check, ελέγχεται αν το πρόγραμμα έχει φτάσει στην αρχή του buffer, μιας και στην συνέλιξη, τα στοιχεία του buffer θα λαμβάνονται από το τέλος προς την αρχή. Στον καταχωρητή A12 αποθηκεύεται κάθε φορά το τέλος του buffer. Οπότε, αν το πρόγραμμα έχει φτάσει στην αρχή του, γίνεται branch στο κομμάτι new_addr, όπου με μία MV εντολή επαναφέρεται ο καταχωρητής A7 στο τέλος του buffer:

```
new_addr:
           MV A12,A7
           B conv
           NOP 5
```

Το υπολογιστικό μέρος γίνεται στο conv, αφού προηγηθεί ο έλεγχος του check:

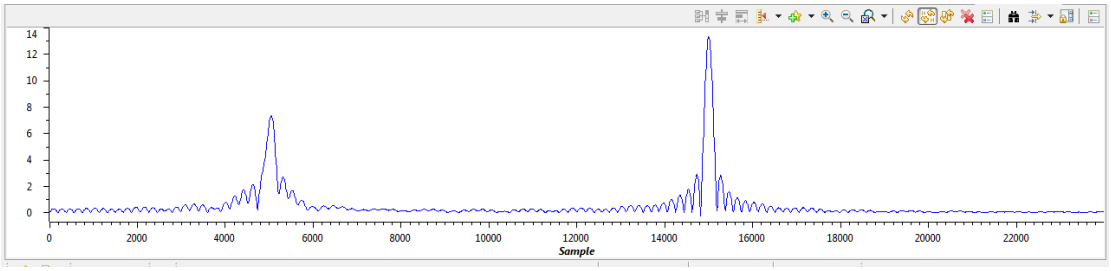
```
conv:
          LDH *--A7,A4
          LDH *+A13[A14],A5
          NOP 4
          MPY A4,A5,A6
          NOP
          SHR A6,15,A6     ;Q-15 FORMAT
          ADD A6,A15,A15   ;ADD RESULT

          ADD A14,1,A14    ;COEFF ELEMENT COUNTER
          SUB B0,1,B0           ;COEFF COUNTER
          [B0]  B check
          NOP 5
          STH A15,*A9++    ;STORE RESULT IN RIGHT BUFFER, THEN
INCREMENT

          B IRP
          NOP 5
```

Φορτώνεται κάθε φορά η τελευταία τιμή του buffer και η πρώτη του φίλτρου. Έπειτα, η προτελευταία του buffer και η δεύτερη του φίλτρου και ούτω καθ' εξής. Ύστερα πολλαπλασιάζονται μεταξύ τους και γίνεται και ένα shift για το Q-15 format. Η τιμή αυτή προστίθεται κάθε φορά σε έναν αθροιστή A15. Μετά, προστίθεται ο αριθμός 1 στον A14 για να πάρει το φίλτρο στην επόμενη επανάληψη, την επόμενη τιμή και αφαιρείται 1 από τον B0 που ελέγχει με αυτόν τον τρόπο αν υπάρχουν ακόμα στοιχεία στο φίλτρο για υπολογισμό. Αν υπάρχουν, επαναλαμβάνεται η διαδικασία, αλλιώς η συγκεκριμένη συνέλιξη λαμβάνει τέλος και αποθηκεύεται η τιμή του αθροιστή στον δεξιό buffer, τον buffer εξόδου. Έπειτα, το πρόγραμμα επιστρέφει στην κανονική του λειτουργία με branch IRP εντολή.

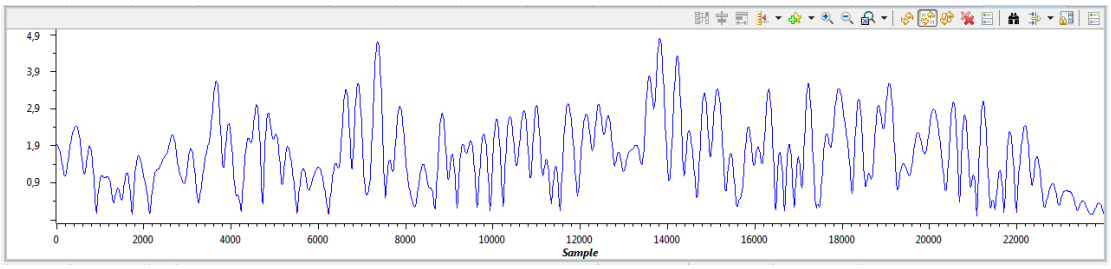
Για να ελεγχθεί η λειτουργία του φίλτρου εφαρμόστηκαν δύο διαφορετικοί τύποι εισόδου, ένας ημιτονοειδής και ένας λευκού θορύβου, και έγινε το FFT διάγραμμα στο CCS.

Είσοδοι:

1. Δύο ημίτονα συχνότητας 5KHz και 15KHz αντίστοιχα.



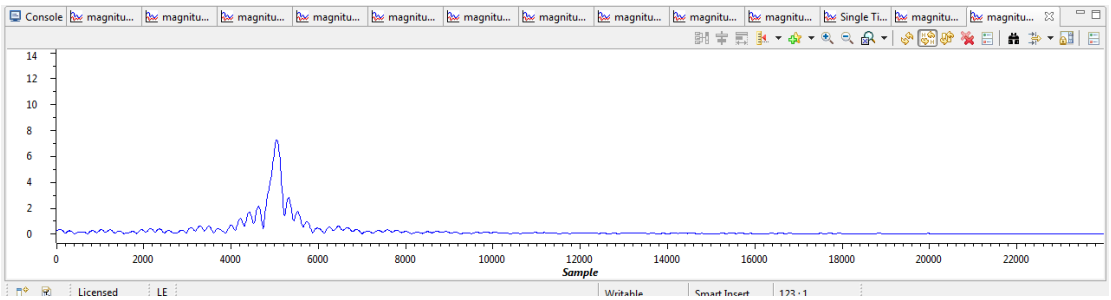
2. Λευκός θόρυβος



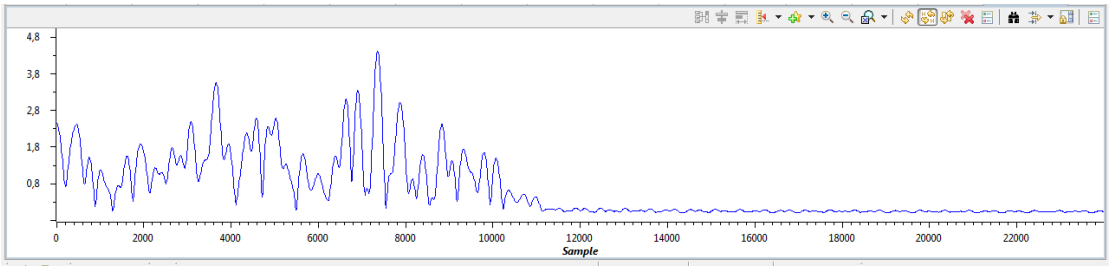
Έξοδοι:

A. Lowpass 1

1. Με είσοδο τα ημίτονα

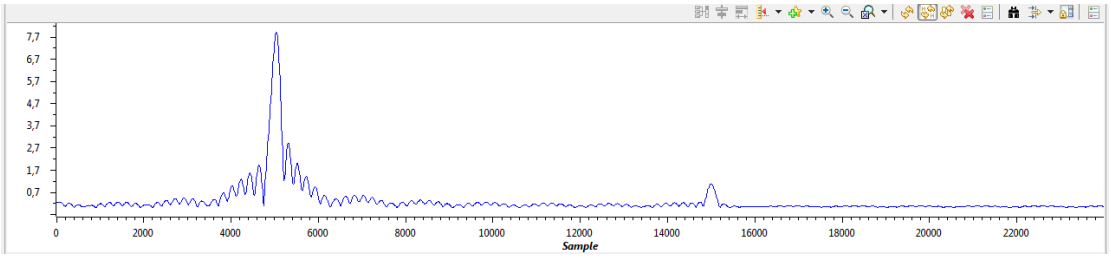


2. Με είσοδο τον λευκό θόρυβο

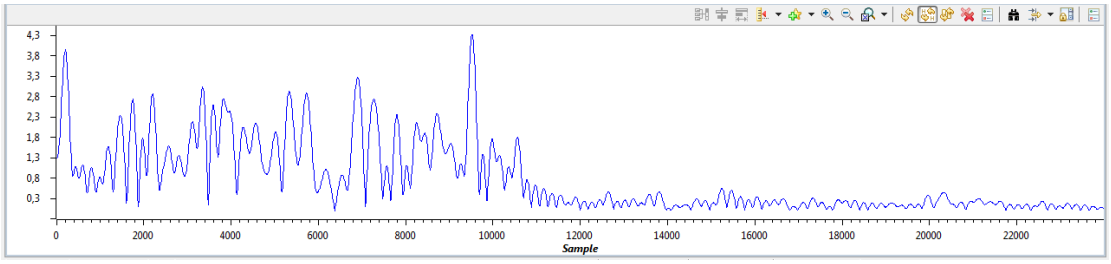


B. Firm Lowpass

1. Με είσοδο τα ημίτονα

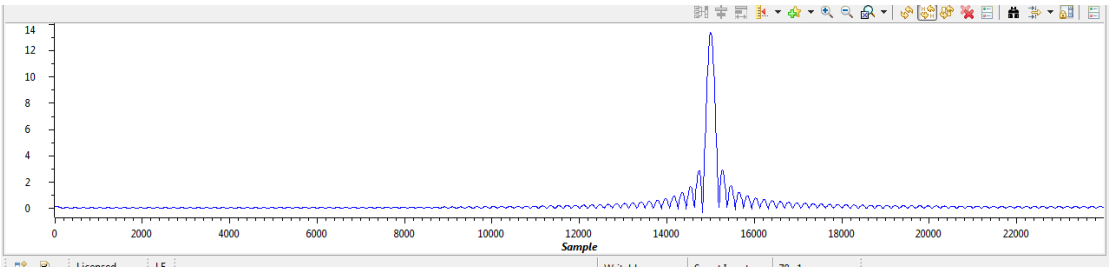


2. Με είσοδο τον λευκό θόρυβο

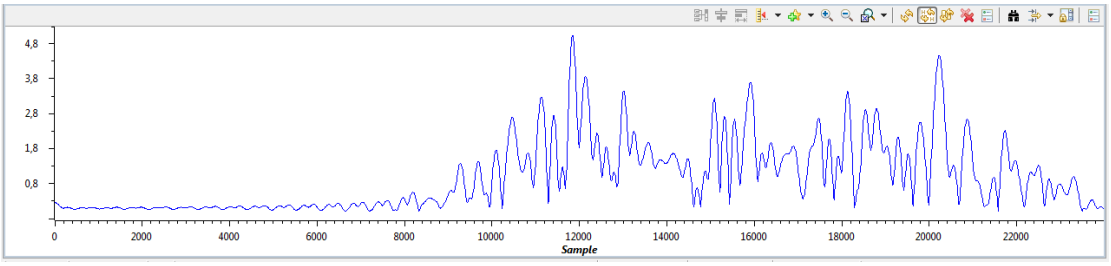


Γ. Highpass 1

1. Με είσοδο τα ημίτονα

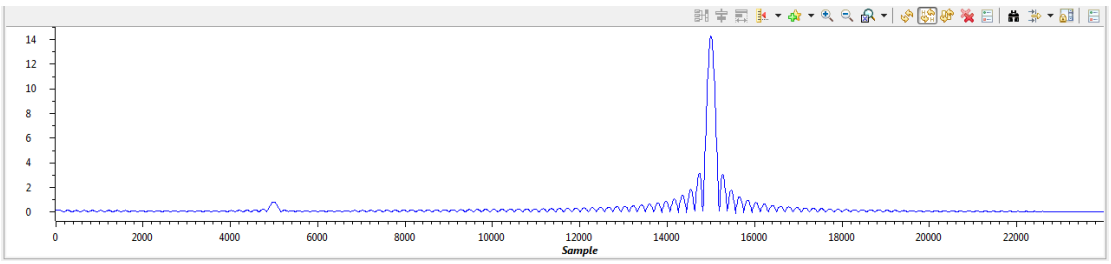


2. Με είσοδο τον λευκό θόρυβο

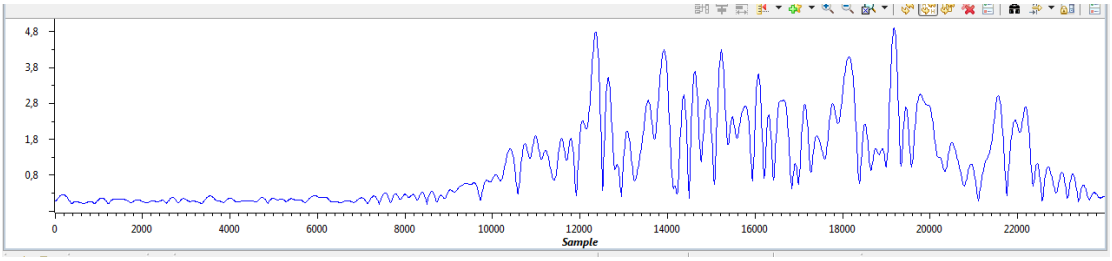


Δ. Firm Highpass

1. Με είσοδο τα ημίτονα

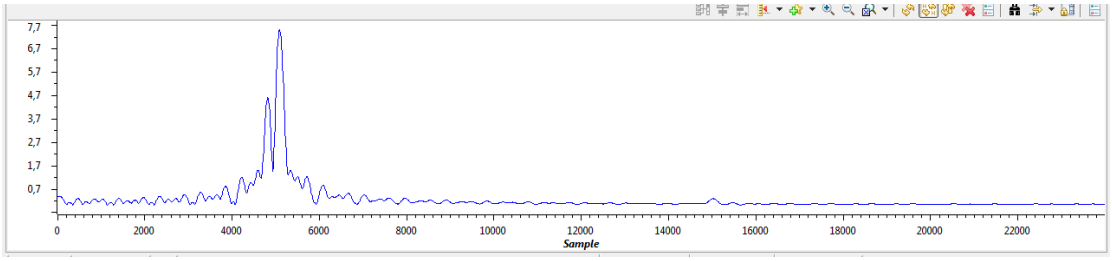


2. Με είσοδο τον λευκό θόρυβο

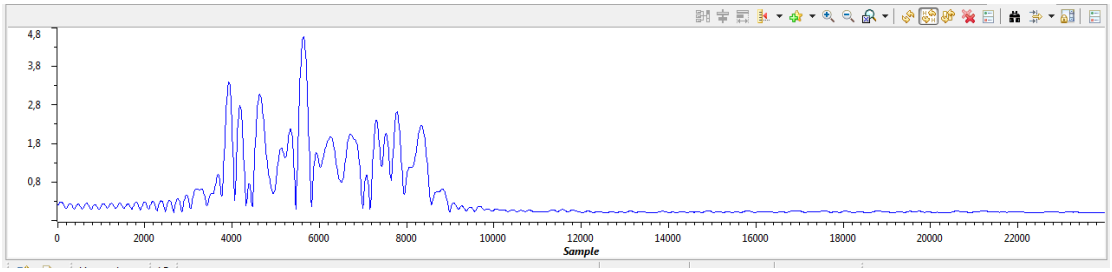


E. Bandpass

1. Με είσοδο 3 ημίτονα, των 1KHz, 5KHz και 15KHz αντίστοιχα.



2. Με είσοδο τον λευκό θόρυβο



Άσκηση 4.8 - Οι μακροεντολές ανάγνωσης και εγγραφής της McBSP:

Να γραφούν οι κατάλληλες μακροεντολές για να γράφετε το περιεχόμενο ενός καταχωρητή στην McBSP1 (transmit) και να διαβάζετε από την McBSP1 (receive) σε έναν καταχωρητή.

Πρόγραμμα:

```
transmit    .macro r1
transmit?:
            STW r1, *A0
            .endm

receive     .macro r2
receive?:
            LDW *A1, r2
            NOP 4
            .endm
```

Άσκηση 4.9 –Σε συνέχεια της άσκησης 4.6:

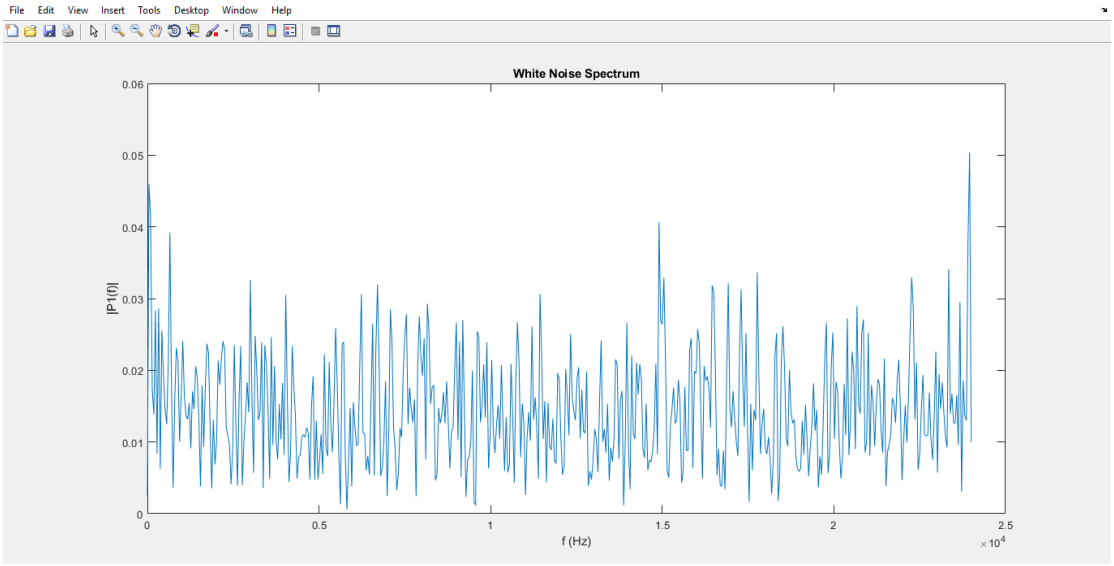
Να αποθηκευθεί το περιεχόμενο του buffer ως αρχείο δεκαεξαδικών (hex file) κάνοντας δεξί κλικ στο Memory Browserκαι επιλέγοντας Save Memoryαφού πρώτα επιλεγεί η προβολή σε “32-bitHex-CStyle”. Το αρχείο που θα αποθηκευτεί θα πρέπει να είναι της μορφής .dat και όχι .bin που είναι η προεπιλογή.. Μετά την εισαγωγή του ονόματος του αρχείου, να εισαχθεί η διεύθυνση του buffer ως buffer_left(ή buffer_right) και το μήκος του (512 bytes). Χρησιμοποιώντας τη συνάρτηση της Matlab "load_vector.m”, να διαβαστεί το δεκαεξαδικό αρχείο που αποθηκεύτηκε, στο MATLAB

Να σχεδιαστεί το σήμα και το φάσμα του (με χρήση της εντολής fftου Matlab), και συγκριθεί με αυτά που λήφθηκαν από το CCS. Να επαναληφθεί το παραπάνω πείραμα χρησιμοποιώντας δείγμα ανθρώπινης φωνής αφού τεθεί ένα μικρόφωνο ως πηγή εισόδου. Να μελετηθεί, από το εγχειρίδιο του AIC23 codec, η δομή των καταχωρητών ελέγχου του και να τροποποιηθεί κατάλληλα το αρχείο “aic23_init.c” ώστε να λειτουργήσει η είσοδος μικροφώνου του board. ΝΑ σχεδιαστεί το φάσμα της φωνής. Σε ποιο φάσμα συχνότητας συγκεντρώνεται η ενέργεια;

Ο κώδικας σε matlab που αναπτύχθηκε για το ερώτημα είναι ο ακόλουθος:

```
load_vector('bufferright.dat');

Y = fft(ans);
Fs = 48000;           % Sampling frequency
T = 1/Fs;             % Sampling period
L = 1024;             % Length of signal
t = (0:L-1)*T;        % Time vector

P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);

f = Fs*(0:(L/2))/L;
plot(f,P1)
title('White Noise Spectrum')
xlabel('f (Hz)')
ylabel('|P1(f)|')
```

Αφού φορτώθηκαν τα δεδομένα και χρησιμοποιήθηκε το αρχείο load vectors του eclass , αρχικοποιήθηκαν οι απαραίτητες συνθήκες έτσι ώστε , μέσω τη εντολής fft να χαραχθεί ο FFT του σήματος που ήταν αποθηκευμένο στον buffer, το οποίο ήταν ένα σήμα λευκού θορύβου.



Άσκηση 4.10 – Κυκλικός buffer:

1.Να μελετηθεί το TMS320C62x/C67x CPU and Instruction Set Reference Guide και να οριστούν οι κυκλικοί buffers.

2. Να τροποποιηθεί ο κώδικας assembly φιλτραρίσματος FIR για να χρησιμοποιηθεί η κυκλική αποθήκευση. 3.Να μετρηθούν οι κύκλοι ρολογιού για κάθε υπολογισμό του FIR φίλτρου.

3.Να βελτιστοποιηθεί ο κώδικας ώστε να μειωθούν όσο το δυνατόν περισσότερο οι κύκλοι ρολογιού

Ο κώδικας που αναπτύχθηκε είναι ο ακόλουθος:

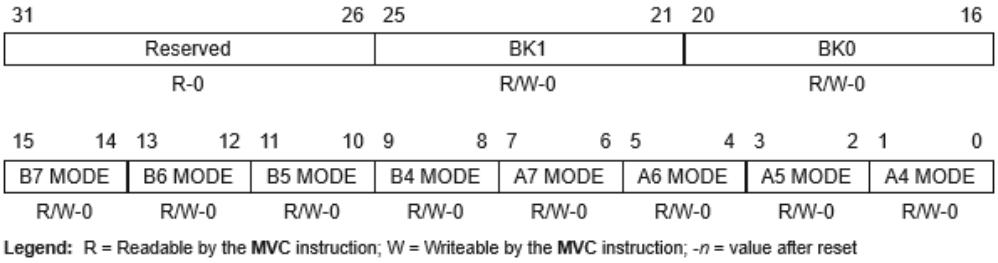
```
            .align 512
SPCR  .set 0x1900008 ; SPCR ADDRESS
DRR   .set 0x1900000 ; DRR ADDRESS
DXR   .set 0x1900004 ; DXR ADDRESS
RCR   .set 0x190000C ; RCR ADDRESS
XCR         .set 0x1900010 ; XCR ADDRESS

IML_pointer      .set 0x019c0004
IMH_pointer      .set 0x019c0000

buffer_left           .space 512
buffer_right     .space 512

            .def _entry
            .def int
```

```
            .ref coef
            .text

_entry:
            MVKL 0xFFFF,B0 ; CLEAR ALL PREVIOUS INTERRUPTS
            MVKH 0xFFFF,B0
            MVC B0,ICR
            ZERO B0

            MVKL  DXR,A0  ; DXR POINTER
            MVKH  DXR,A0

            MVKL  DRR,A1 ;DRR POINTER
            MVKH  DRR,A1

            MVKL IML_pointer, A3 ; IML POINTER
            MVKH IML_pointer, A3

            LDW *A3,A5  ; RINT1 -> INT4
            NOP 4
            SET A5,0,3,A5
            STW A5,*A3
            ZERO A5

            MVKL  RCR,A5     ; RCR POINTER
            MVKH  RCR,A5

            MVKL  XCR,A6     ; XCR POINTER
            MVKH  XCR,A6

            MVKL  0xA0,A7    ; 0xA0
            MVKH  0xA0,A7

            STW        A7,*A5  ;SET RCR, XCR
            STW     A7,*A6

            ZERO A5
            ZERO A6
            ZERO A7

            MVKL 0X13,B1 ;SET  NMI AND INT4 TO 1
            MVC B1,IER

            MVKL 1,B2 ;SET GIE TO 1
            MVC B2,CSR

            ZERO B1
            ZERO B2

            MVKL 0x1082500, B7    ;AMR
            MVKL 0x1082500, B7
            MVC B7, AMR

            MVK 0x100, B2 ;COUNTER = 100

            MVKL coef,A13 ; COEFF ADRESS
```

```
           MVKH coef,A13

loop:
           B loop
           NOP 5

int:
           LDW *A1,A4
           NOP 4
           MV B4, B5

           ZERO A14
           ZERO A15

           STH A4,*B4++     ; LEFT BUFFER STORE (INPUT)

           MVKL 0x28,B0
           MVKH 0x28,B0

conv:
           LDH *--B5,A4
           LDH *+A13[A14],A5
           NOP 4
           MPY A4,A5,A6
           NOP
           SHR A6,15,A6     ;Q-15 FORMAT
           ADD A6,A15,A15   ;ADD RESULT

           ADD A14,1,A14    ;COEFF ELEMENT COUNTER
           SUB B0,1,B0           ;COEFF COUNTER
           [B0]  B conv
           NOP 5
           STH A15,*B6++    ;STORE RESULT IN RIGHT BUFFER, THEN
INCREMENT

           B IRP
           NOP 5
```

Η υλοποίηση του κυκλικού buffer απλοποιεί το πρόγραμμα καθώς δεν απαιτείται πλέον το κομμάτι του κώδικα που ήλεγχε τον διαθέσιμο χώρο που υπάρχει στον buffer καθώς και τα κομμάτια που αφορούν την αρχικοποίηση του buffer σε περίπτωση που είναι γεμάτος. Οι παραπάνω έλεγχοι γίνονται αυτόματα μέσω της κυκλικότητας.

Προκειμένου να οριστεί ένας κυκλικός buffer χρησιμοποιείται ο καταχωρητής AMR ο οποίος έχει μέγεθος 32 bit. Τα 16 LSB χρησιμοποιούνται για να επιλέξουν τον τρόπο λειτουργίας για κάθε έναν από τους 8 επιτρεπόμενους καταχωρητές A4-A7,B4-B7 και τα 10 υψηλότερα ορίζουν το μέγεθος του buffer N.



31–26	25–21	20–16
Reserved	BK1	BK0
R-0	R/W-0	R/W-0

15–14	13–12	11–10	9–8	7–6	5–4	3–2	1–0
B7 MODE	B6 MODE	B5 MODE	B4 MODE	A7 MODE	A6 MODE	A5 MODE	A4 MODE
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Readable by the **MVC** instruction; W = Writeable by the **MVC** instruction; *-n* = value after reset

Οι καταχωρητές που χρησιμοποιούνται στην προκειμένη περίπτωση είναι οι B4 ,B5 που δείχνουν στο μπλοκ BK0 και ο B6 που δείχνει στο BK1. Συνεπώς πρέπει τα bit 8 και 10 ,που αφορούν τους B4 ,B5 να λάβουν την τιμή 1 Hex και τα bit 12 ,13 να λάβουν την τιμή 2 Hex σύμφωνα με τον πίνακα που επεξηγεί τις τιμές του κάθε bit.

Bit	Field	Value	Description
13–12	B6 MODE	0–3h	Address mode selection for register file B6.
		0	Linear modification (default at reset)
		1h	Circular addressing using the BK0 field
		2h	Circular addressing using the BK1 field
		3h	Reserved
11–10	B5 MODE	0–3h	Address mode selection for register file B5.
		0	Linear modification (default at reset)
		1h	Circular addressing using the BK0 field
		2h	Circular addressing using the BK1 field
		3h	Reserved
9–8	B4 MODE	0–3h	Address mode selection for register file B4.
		0	Linear modification (default at reset)
		1h	Circular addressing using the BK0 field
		2h	Circular addressing using the BK1 field
		3h	Reserved

Το μέγεθος N που γράφεται στα bits 16-20,21 -25 υπολογίζεται από τον buffer_size= 2^{N+1} bytes. Επομένως 512= 2^{N+1}) => N=8. Συνεπώς ο αριθμός που γράφεται στον AMR είναι 0x1082500 στον register B7.

Αφού γίνουν οι αρχικοποιήσεις των απαραίτητων καταχωρητών, ακολουθεί το interrupt (int) στο οποίο φορτώνεται το περιεχόμενο του DRR και αποθηκεύεται στον buffer μέσω του pointer B4 ο οποίος περιέχει της διεύθυνση αρχής του buffer.Η διεύθυνση πρώτα αποθηκεύεται και στον B5 μέσω της εντολής MV B4 , B5. Στη συνέχεια ο pointer B4 αυξάνει κατά ένα την τιμή του έτσι ώστε το κάθε νέο στοιχείο που έρχεται να αποθηκεύεται στην αμέσως επόμενη θέση του buffer εισόδου. Τέλος γίνεται η αρχικοποίηση του μετρητή B0 για την συνέλιξη.

```
int:
            LDW *A1,A4
            NOP 4
            MV B4, B5

            ZERO A14
            ZERO A15

            STH A4,*B4++     ; LEFT BUFFER STORE (INPUT)

            MVKL 0x28,B0
            MVKH 0x28,B0
```

Έπειτα, ακολουθεί ο βρόχος της στον οποίο γίνεται η συνέλιξη.

```
conv:
            LDH *--B5,A4
            LDH *+A13[A14],A5
            NOP 4
            MPY A4,A5,A6
            NOP
            SHR A6,15,A6     ;Q-15 FORMAT
            ADD A6,A15,A15   ;ADD RESULT

            ADD A14,1,A14    ;COEFF ELEMENT COUNTER
            SUB B0,1,B0            ;COEFF COUNTER
            [B0]  B conv
            NOP 5
            STH A15,*B6++    ;STORE RESULT IN RIGHT BUFFER, THEN
                             ;INCREMENT

            B IRP
            NOP 5
```

Στον βρόχο αυτόν , γίνεται ο πολλαπλασιασμός του τελευταίου στοιχείου του buffer με το πρώτο στοιχείο των coeffs , του προ τελευταίου με το δεύτερο στοιχείο των coeffs κοκ. Αυτό επιτυγχάνεται μέσω των pointers B5 και A13 οι οποίοι περιέχουν τις διευθύνσεις αρχής του κυκλικού buffer και των coeffs αντίστοιχα. Ο buffer διατρέχεται προς τα πίσω , για αυτό και χρησιμοποιείται ο τελεστής -- , ενώ τα coeffs διατρέχονται κανονικά. Προκειμένου να επιλέγεται το σωστό στοιχείο των coeffs κάθε φορά χρησιμοποιείται ο καταχωρητής A14 σαν μετρητής. Ο A14 ξεκινάει με τιμή 0 και μετά από κάθε πολλαπλασιασμό αυξάνει την τιμή του κατά 1. Αφού φορτωθούν τα σωστά στοιχεία σε κάθε καταχωρητή γίνεται ο πολλαπλασιασμός και ένα shift right για το Q 15 format. Τα στοιχεία αποθηκεύονται στον αθροιστή A15 και οι παραπάνω διαδικασία επαναλαμβάνεται μέχρι ο register B0 να γίνει 0. Μόλις ο B0 λάβει την τιμή 0 τα δεδομένα αποθηκεύονται στον δεύτερο κυκλικό buffer που ορίστηκε , η διεύθυνση του οποίου είναι αποθηκευμένη στον B6. Ο B6 αποθηκεύει τα

αποτελέσματα στις διαδοχικές θέσεις του buffer και με το τέλος της συνέλιξης γίνεται ένα Branch στον IRP.

Βιβλιογραφία:

- **TMS320C67x/C67x+ DSP CPU and Instruction Set Reference Guide**
- **ΕΚΦΩΝΗΣΗ ΑΣΚΗΣΗΣ LAB-4 – ECLASS**
- **TMS320C6713 DATASHEET**