ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ

***Ανάπτυξη αλγόριθμων ψηφιακής επεξεργασίας σήματος σε πραγματικό χρόνο, ψηφιακών φίλτρων IIR και FIR σε πλακέτα* TMS320C6713 DSK Κωδικός :09170ΥΣ**

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Κυριάκος Κυριάκου του Κωνσταντίνου
Κ.Α.Σ:504701

Επιβλέπων Καθηγητής: ΣΩΤΗΡΙΟΣ ΠΟΥΡΟΣ

Θεσσαλονίκη, Φεβρουάριος 2010

Ονοματεπώνημο:Κυριάκος Κυριάκου του Κωνσταντίνου

Κ.Α.Σ:504701

Επιβλέπων Καθηγητής: ΣΩΤΗΡΙΟΣ ΠΟΥΡΟΣ

Ηλεκτρονικός Τ.Ε.Ι.Θ.

M.Sc. in Data Telecommunications and Networks, Salford University

Ημερομηνία ανάληψης:23/2/2009

Ημερομηνία παράδοσης :22/1/2010

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΕΧΟΜΕΝΑ

Πρόλογος

Η παρούσα πτυχιακή εργασία Ανάπτυξη αλγόριθμων ψηφιακής επεξεργασίας σήματος σε πραγματικό χρόνο, ψηφιακών φίλτρων IIR και FIR σε πλακέτα TMS320C6713 DSK πραγματοποιήθηκε κατά το Ακαδημαϊκό έτος 2009-2010 στο εργαστήριο Γ2 του κ.Κυρτόπουλλου Σταύρου στο Τμήμα Ηλεκτρονικής του Α.Τ.Ε.Ι Θεσσαλλονίκης .

Θα ήθελα να ευχαριστήσω τον καθηγητή κ.Σωτήριο Πούρο που ήταν ο θεματοδότης και επιβλέπων της πτυχιακής μου εργασίας, για την σημαντική και ουσιαστική βοήθεια που μου παρείχε.

ΠΕΡΙΛΗΨΗ

Βασικός σκοπός αυτής της πτυχιακής εργασίας είναι η μελέτη ,η σχεδίαση και η εφαρμογή των ψηφιακών φίλτρων FIR(Finite Impulse Response) και IIR(Infinite Impulse Response) σε πλακέτα TMS320C6713 DSK.

Στο κεφάλαιο 1 γίνεται μια εισαγωγή στα αναλογικά φιλτρα και ψηφιακά φίλτρα και βλέπουμε τις διαφορές των αναλογικών και ψηφιακών φιλτρων .

Στο κεφάλαιο 2 παρουσιάζεται η εξήγηση της λειτουργίας των ψηφιακών φίλτρων FIR και IIR.

Στο κεφάλαιο 3 γίνεται εισαγωγή στη πλακέτα TMS320C6713 DSK και στο περιβάλλον του Code Composer Studio.

Στο κεφάλαιο 4 παρουσιάζεται ο υπολογισμός αλγόριθμων IIR φίλτρων με το fdatool και Matlab,και ολοκλήρωση προγράμματος C και κατέβασμα στο DSP μέσω του code composer studio.

Στο κεφάλαιο 5 παρουσιάζεται ο υπολογισμός αλγόριθμων FIR φίλτρων με το fdatool και Matlab,και ολοκλήρωση προγράμματος C και κατέβασμα στο DSP μέσω του code composer studio.

Στο κεφάλαιο 6 αναλύεται ο υπολογισμός της συνάρτησης μεταφοράς για FIR και IIR φίλτρα.

Στο κεφάλαιο 7 παρουσιάζεται η υλοποίηση των πειραμάτων FIR.

Στο κεφάλαιο 8 παρουσιάζεται η υλοποίηση των πειραμάτων IIR.

Summary

Basic aim of this final work is the study, the designing and application of digital filters FIR (Finite Impulse Response) and IIR (Infinite Impulse Response) in plaques TMS320C6713 DSK.

In capital 1 becomes a import in the proportional filters and digital filters and sees the differences the proportional and digital filters.

In capital 2 are presented the explanation of operation of digital filters FIR and IIR.

In capital 3 becomes import in plaque TMS320C6713 DSK and in the environment of Code Composer Studio.

In capital 4 are presented the calculation of algorithms IIR of filters with fdatool and Matlab, and completion of program C and downshift in the DSP via code composer studio.

In capital 5 are presented the calculation of algorithms FIR of filters with fdatool and Matlab, and completion of program C and downshift in the DSP via code composer studio.

In capital 6 is analyzed the calculation of interelation of transport for FIR and IIR filters.

In capital 7 is presented the concretisation of experiments FIR

In capital 8 is presented the concretisation of experiments IIR

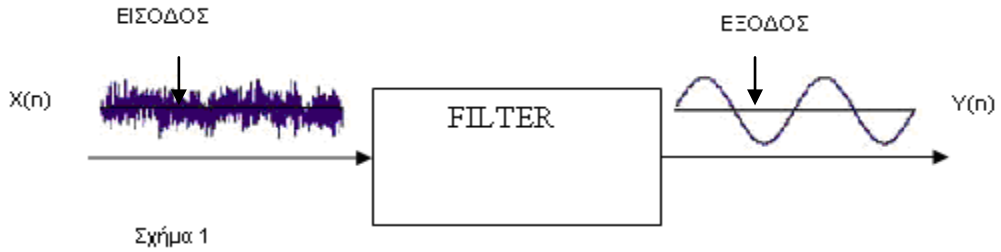
ΚΕΦΑΛΑΙΟ 1

Εισαγωγή στα ψηφιακά φίλτρα

1.1 Τι είναι φίλτρο

Σαν φίλτρο ορίζουμε το σύστημα εκείνο που επεξεργάζεται κάποιο σήμα με σκοπό την ανίχνευση και τον διαχωρισμό ενός επιθυμητού σήματος από το ανεπιθύμητο σύνολο θορύβων, ή να εξάγει επιθυμητά μέρη ενός σήματος .

Η συμπεριφορά ενός φίλτρου συνήθως περιγράφετε από τη σχέση εισόδου εξόδου.



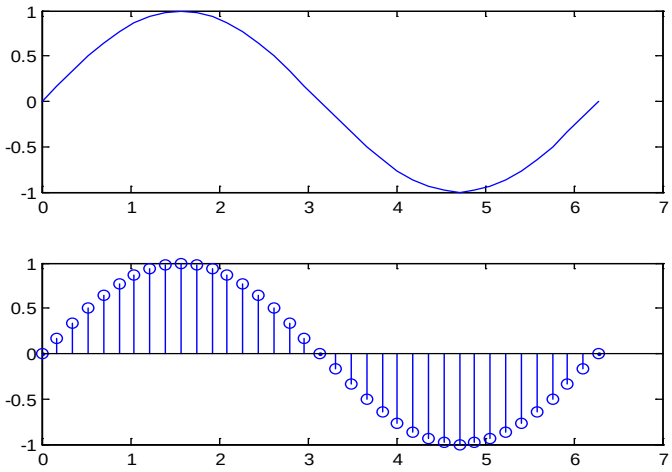
Σχήμα 1

1.2Είδη φίλτρων

Υπάρχουν δύο είδη φίλτρων, τα αναλογικά και τα ψηφιακά.

Είναι δύο διαφορετικά φίλτρα τα οποία διαφέρουν στην λειτουργία και στην κατασκευή.

- Αν επεξεργαζόμαστε αναλογικά σήματα τότε τα φίλτρα λέγονται αναλογικά .
- Αν επεξεργαζόμαστε ψηφιακά σήματα τότε τα φίλτρα είναι ψηφιακά .



Σχήμα 2 Αναλογικό και ψηφιακό σήμα .

1.2.1 Αναλογικά φίλτρα

Το αναλογικό φίλτρο χρησιμοποιεί αναλογικά ηλεκτρονικά κυκλώματα κατασκευασμένα από υλικά κυρίως αντιστάσεις, πυκνωτές και τελεστικούς ενισχυτές για να παράγει το ζητούμενο αποτέλεσμα φιλτραρίσματος.

Τα κυκλώματα αναλογικών φίλτρων χρησιμοποιούνται για :

- ελάττωση θορύβου
- βελτίωση σήματος βίντεο
- γραφικούς αναλυτές
- συστήματα ήχου

1.2.1.1 Κατηγορίες αναλογικών φίλτρων

Α. Παθητικά αναλογικά φίλτρα

Τα παθητικά φίλτρα ονομάζονται και φίλτρα RLC. Τα παθητικά φίλτρα είναι κατασκευασμένα από συνδυασμό στοιχείων δύο ακροδεκτών τα οποία είναι αντιστάσεις, πηνία και πυκνωτές .

Τα στοιχεία αυτά (αντίσταση, πηνίο και πυκνωτής) ονομάζονται παθητικά στοιχεία, για αυτό τα φίλτρα που είναι κατασκευασμένα από αυτά τα στοιχεία ονομάζονται παθητικά φίλτρα, τα οποία απορροφούν ισχύ. Όταν συνδεθούν σ' ένα κύκλωμα πρέπει να τους δοθεί ενέργεια για να λειτουργήσουν σε αντίθεση με άλλα στοιχεία (ενεργά) π.χ. οι ενισχυτές ή μπαταρίες, οι οποίοι δίνουν ενέργεια .

Βασικό μειονέκτημα των παθητικών φίλτρων είναι ότι δεν έχουν προσαρμογή.

Κάθε φίλτρο έχει μία είσοδο και μία έξοδο. Για να υπολογισθεί το νέο σήμα στην έξοδο ενός φίλτρου, που αποτελείται από αντιστάσεις, πηνία, πυκνωτές ή συνδυασμό αυτών, θα πρέπει να υπολογιστούν οι τιμές αυτών των στοιχείων που αποτελούν το κύκλωμα. Το φίλτρο δεν δουλεύει από μόνο του αλλά συνήθως παρεμβάλλεται μέσα σε άλλα κυκλώματα. (Πριν απ' το φίλτρο σίγουρα θα υπάρχει κάποιο κύκλωμα και μετά απ' το φίλτρο θα έπεται κάποιο άλλο κύκλωμα). Η συνθήκη αυτή, το να υπεισέρχεται δηλαδή η αντίσταση εισόδου κι αντίσταση εξόδου στον υπολογισμό του φίλτρου, σημαίνει ότι όταν το φίλτρο θα μπει στη θέση του θα πρέπει να προσαρμόζεται σωστά. Θα πρέπει η αντίσταση εισόδου του φίλτρου να ταιριάζει με την αντίσταση εξόδου του κυκλώματος που προηγείται και αντίστοιχα η αντίσταση εξόδου του φίλτρου να ταιριάζει με την αντίσταση εισόδου του κυκλώματος που έπεται. Αυτό το ταίριασμα των αντιστάσεων ονομάζεται προσαρμογή.

Β. Ενεργά φίλτρα

Τα παθητικά φίλτρα, όπως προαναφέρθηκε, έχουν το μειονέκτημα της προσαρμογής. Για να αποφύγουμε αυτό το πρόβλημα και για να αποποιηθούν τα φίλτρα, κατασκευάζουμε ενεργά φίλτρα.

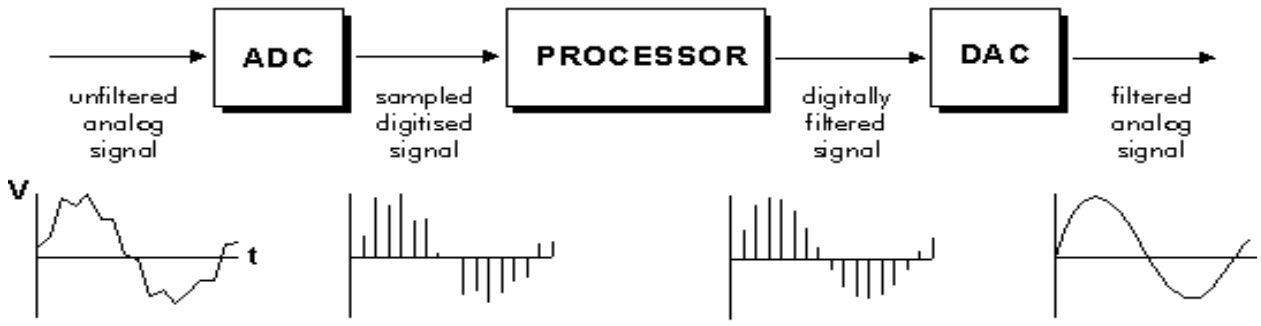
Για να λύσουν το πρόβλημα της προσαρμογής σε ένα παθητικό φίλτρο, προσθέτουμε ένα τελεστικό ενισχυτή μετά από αυτό. Η λειτουργία του τελεστικού ενισχυτή δεν είναι για να ενισχύσει κάτι, αλλά για να απομονώσει το φίλτρο απ' το φορτίο.

Αν μετά τον ενισχυτή μπαίνει κάποιο μεταβλητό φορτίο (το οποίο είναι το κύκλωμα που ακολουθεί), ο ίδιος καταφέρνει αυτό το φορτίο να μην επηρεάζει το φίλτρο και να έχουμε σωστή προσαρμογή. Θα δούμε ότι η ενίσχυση αυτών των ενισχυτών είναι μονάδα, δηλαδή ότι παίρνουν στην είσοδο, το ίδιο βγάζουν στην έξοδο και γι' αυτόν τον λόγο, επειδή η εφαρμογή τους είναι ειδική, ονομάζονται απομονωτές ή buffer.

1.2.2 Ψηφιακά φίλτρα

Με την ανάπτυξη τις τεχνολογίας και των ψηφιακών συστημάτων άρχισαν να αναπτύσσονται τεχνικές για την επεξεργασία και φιλτράρισμα διακριτών σημάτων (ψηφιακών)που τα σήματα αυτά έρχονται από την δειγματοληψία αναλογικών σημάτων. Ένα ψηφιακό φίλτρο υλοποιείται με ένα ψηφιακό επεξεργαστή μέσο ηλεκτρονικού υπολογιστή .

Το αναλογικό σήμα εισόδου πρέπει πρώτα να δειγματοληπτεί και να ψηφιοποιηθεί με τη χρήση ενός ADC (analogy to digital converter) μετατροπέα από αναλογικό σε ψηφιακό σήμα . Οι δυαδικοί αριθμοί που προκύπτουν, οι οποίοι αναπαριστούν διαδοχικές τιμές από τη δειγματοληψία του σήματος εισόδου μεταφέρονται στον επεξεργαστή, που εκτελεί αριθμητικές πράξεις και μετά ξαναμετατρέπεται από ένα DAC μετατροπέα από ψηφιακό σε αναλογικό.



Σχήμα (1.2.2) Λειτουργία DSP

1.3 Διαφορές ψηφιακών και αναλογικών φίλτρων

1. Ένα ψηφιακό φίλτρο είναι σε θέση να προγραμματιστεί, όταν η λειτουργία του καθορίζεται από ένα πρόγραμμα στη μνήμη ενός επεξεργαστή. Αυτό σημαίνει ότι το ψηφιακό φίλτρο μπορεί να αλλαχθεί εύκολα χωρίς να επηρεαστεί το κύκλωμα.
2. Ένα αναλογικό φίλτρο, μπορεί μόνο να αλλαχθεί ξανασχεδιάζοντας το κύκλωμα του φίλτρου δηλαδή πρέπει να αλλάξουμε τα ηλεκτρικά του εξαρτήματα .
3. Οι χαρακτηριστικές των αναλογικών φίλτρων είναι συνάρτηση τον τιμών των στοιχείων των ηλεκτρικών κυκλωμάτων έτσι που οι τιμές τους αλλάζουν με την θερμοκρασία , υγρασία , γήρανση. Ως αποτέλεσμα να μεταβάλλονται οι χαρακτηριστικες των φίλτρων .
4. Στα ψηφιακά είναι απόλυτος σταθερές οι χαρακτηριστικές αν δουλεύει ο επεξεργαστής κανονικά .
5. Με τα ψηφιακά προσεγγίζουμε τις ιδανικές τιμές ενός φίλτρου ,ενώ στα αναλογικά δεν έχουμε ακρίβεια στις τιμές γιατί απαιτούνται πολύπλοκα κυκλώματα λόγο ότι είναι δύσκολος ο υπολογισμός τον στοιχείων του κυκλώματος.
6. Τα ψηφιακά φίλτρα έχουν πολύ μεγάλη ευελιξία στο σχεδιασμό τους ενώ τα αναλογικά όχι.

1.4 Κατηγορίες ψηφιακών φίλτρων

Τα ψηφιακά φίλτρα είναι συστήματα που ενεργούν πάνω στα ψηφιακά σήματα με σκοπό να αλλάξουν τα χαρακτηριστικά τους όπως το φάσμα και το πλάτος. Ο κυριότερος σκοπός τους είναι να βελτιωθεί η ποιότητα του σήματος με την ανάδειξη συγκεκριμένων χαρακτηριστικών ή με την εξαγωγή του θορύβου.

Τα γραμμικά ψηφιακά φίλτρα μπορούν να υλοποιηθούν με την βοήθεια λογισμικού ή υλικού (software ή hardware) και αποτελούν το ουσιαστικότερο εργαλείο στην ψηφιακή επεξεργασία σήματος.

Τα γραμμικά ψηφιακά φίλτρα χωρίζονται σε δύο μεγάλες κατηγορίες οι οποίες χαρακτηρίζονται από το μήκος της κρουστικής απόκρισης του φίλτρου.

Τα φίλτρα που έχουν πεπερασμένη κρουστική απόκριση ονομάζονται FIR (Finite Impulse Response) ενώ αυτά με μη-πεπερασμένη κρουστική απόκριση ονομάζονται IIR (Infinite Impulse Response).

Τα FIR φίλτρα ονομάζονται και μη αναδρομικά φίλτρα επειδή η τρέχουσα έξοδος T.{y(n)} υπολογίζεται μεμονωμένα από τις τρέχουσες και προηγούμενες εισαγόμενες τιμές{x(n), x(n-1), x(n-2), ...}.

Μια εφαρμογή FIR φίλτρου φαίνετε πιο κάτω:

y(n) = x(n) - x(n-1)

Η εξαγόμενη τιμή σε t = nh είναι ίση με τη διαφορά ανάμεσα στον τρέχοντα εισαγόμενο χρόνο xn και στον προηγούμενο εισαγόμενο x(n-1):

y(0) = x(0) – x(-1)

y(1) = x(1) – x(0)

y(2) = x(2) – x(1)

y(3) = x(3) – x(2)

... κ.λ.π.

Το εξαγόμενο είναι η αλλαγή του εισαγόμενου βάσει των περισσότερο πρόσφατων εισαγωγών.

Τα IIR φίλτρα ονομάζονται και αναδρομικά φίλτρα επειδή περιέχει τις εισαγόμενες τιμές {x(n), x(n-1), x(n-2), ..}αλλά επίσης και προηγούμενες εξαγόμενες {y(n-1), y(n-2), ...}

Μια εφαρμογή IIR φίλτρου φαίνετε πιο κάτω:

y(n) = x(n) + y(n-1)

Το φίλτρο αυτό καθορίζει το τρέχον εξαγόμενο(yn) προσθέτοντας το τρέχον εισαγόμενο (xn) στο προηγούμενο εξαγόμενο (yn-1).

y(0) = x(0) + y(-1)

y(1) = x(1) + y(0)

y(2) = x(2) + y(1)

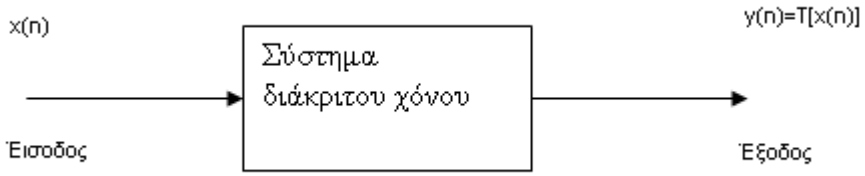
y(3) = x(3) + y(2)

... κ.λ.π.

ΚΕΦΑΛΑΙΟ 2
Ψηφιακά φίλτρα FIR ΚΑΙ IIR

2. Ψηφιακά φίλτρα FIR (Finite Impulse Response).

Ένα σύστημα διάκριτου χρόνου είναι μια υπολογιστική διαδικασία που δέχεται στην είσοδο μια σειρα αριθμών x(n)και τα μετασχηματίζει μια σειρά αριθμών y(n)που είναι η έξοδος .



Σχήμα 2.1 Διάγραμμα συστήματος διάκριτου χρονου

Η λειτουργεια του βασίζεται απο την σχέση y(n)=T[x(n)] η οποία δίνει την έξοδο, όταν μια συνάρτηση T[ ] εφαρμόζετε στα δείγματα εισόδου x[n].

Καθώς τα συστήματα διάκριτου χρονου είναι μια σειρά από αριθμούς, αυτές οι διεργασίες περιγράφονται όταν γνωρίζουμε την συνάρτηση T[ ].

Η γενική σχέση που περιγράφει ένα FIR φίλτρο φαίνεται στην εξίσωση (2.1), όπου το y(n) είναι η έξοδος του φίλτρου, x(n) τα δείγματα εισόδου και b(k) οι συντελεστές του φίλτρου.

$$y(n) = \sum_{K=0}^{N-1} b[k]x[n-k] \qquad \textbf{(2.1)}$$

2.1 Βασικές συναρτήσεις

Οι τρεις βασικές συναρτήσεις που εξασφαλίζουν έναν πληρη χαρακτηρισμό του φίλτρου είναι:

1. Μοναδιαία κρουστική απόκριση δ(n) (The unit impulse sequence)
2. Πεπερασμένη κρουστική απόκριση h(n) (Unit Impulse Response Sequence)
3. Συνέλιξη (Convolution)

2.1.1 Η μοναδιαια κρουστική ακολουθία (unit impulse sequence)

Η μοναδιαια κρουστική ακολουθία είναι η πιο απλή συνάρτηση διότι έχει μόνο μια μη μηδενική τιμή όταν n=0.

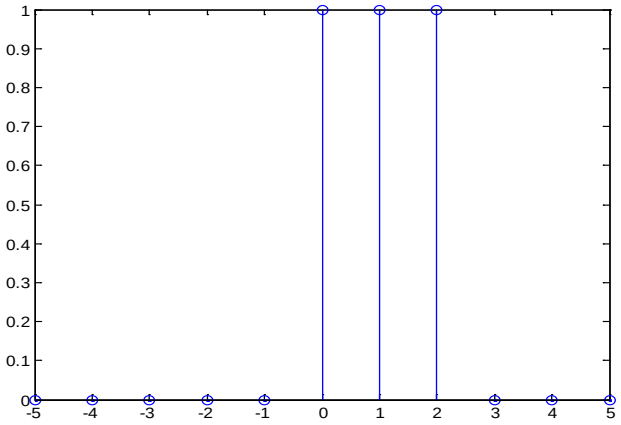
$$\delta(n)= 1 \quad n=0$$

$$\delta(n)= 0 \quad n \neq 1$$

Μια εφαρμογή είναι:

$$X(n)=\delta(n)+\delta(n-1)+\delta(n-2) \quad \textbf{(2.2)}$$

$$x(n) = \sum x(k)\delta(n-k) = ... + x(0)\delta(n) + x(1)\delta(n-1) + x(2)\delta(n-2)...$$



Σχήμα 2.2 Μοναδιαία κρουστική απόκριση

Αποτέλεσμα:

x =0 0 0 0 0 1 1 1 0 0 0

Βλέπουμε οτι δίνει αποτελέσματα όταν το k τρέχει στις μη μηδενικές τιμές της δ[n]. Τα δείγματα μεταφέρονται κατά μια θέση δεξιά και κατα μια θέση αριστερά αν είχαμε δ(n+1.2.3....).

2.1.2 Κρουστική απόκριση h[n].

Όταν η είσοδος σε ένα FIR φίλτρο είναι η μοναδιαια κρουστική ακολουθία **X(n)=δ(n)+δ(n-1)+δ(n-2),** η έξοδος ονομάζεται κρουστική απόκριση πεπερασμένου μήκους και συμβολίζετε με h[n], δηλαδή:

$$(2.2)\, y(n) = h(n) = \sum_{K=0}^{N-1} b[k]\, x[n-k]$$

n	n>0	0	1	2	3	4	N-1
x(n)=δ(n)	0	1	1	1	0	0	0
y(n)=h(n)	0	b(0)	b(1)	b(2)	b(3)	b(4)	b(N-1)

Η κρουστική απόκριση h[n]ενός FIR φίλτρου είναι οι συντελεστές b(k). Καθώς h[n]=0 για n<0 και n>N-1, το μήκος της h[n] και συνεπώς της εξόδου του φίλτρου είναι πεπερασμένο,δηλαδή συγκεκριμένο.

2.1.3 Συνέλιξη

Η γενική εξίσωση των FIR φίλτρων (σχήμα 2.1) μπορεί να χωριστεί σε όρους που αποτελούνται από την μοναδιαία κρουστική απόκριση δ[n]. Υποθέτοντας ότι οι σταθερές του φίλτρου είναι όμοιες με την κρουστική απόκριση h[n] μπορούμε να αντικαταστήσουμε τους b[k] με τα h[n] στην σχέση 2.1 και έχουμε την σχέση:

$$(2.3)\, y(n) = \sum_{K=0}^{N-1} h[k]x[n-k] \quad \textbf{(2.3)}$$

Μία εφαρμογή συνέληξης φαίνεται στο πιο κάτω πίνακα. Παραουσίαζεται η η συνάθροιση των x[n]={6, 4,2} με τα h[n]={4, -2, 2, 1}.

n	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
x(n)				**6**	**4**	**2**			
h(n)	**4**	**-2**	**2**	**1**					
h(0)x(n-0)	**1**	**2**	**-2**	**4**					
h(0)x(n-1)		**1**	**2**	**-2**	**4**				
h(0)x(n-2)			**1**	**2**	**-2**	**4**			
h(0)x(n-3)				**1**	**2**	**-2**	**4**		
h(0)x(n-4)					**1**	**2**	**-2**	**4**	
h(0)x(n-5)						**1**	**2**	**-2**	**4**
y(n)				**24**	**-4**	**12**	**10**	**8**	**2**

Πίνακας 2.1

2.2 Χαρακτηριστικά και είδη FIR φίλτρων

Τα FIR φίλτρα, όπως προαναφέρθηκε, είναι τα ψηφίακα φίλτρα που έχουν πεπερασμένη χρονική διάρκεια όταν στην είσοδο τους εφαρμοστεί μοναδιαία κρουστική απόκριση δ(n).

Υπάρχουν τέσσερα βασικά είδη FIR φίλτρων :

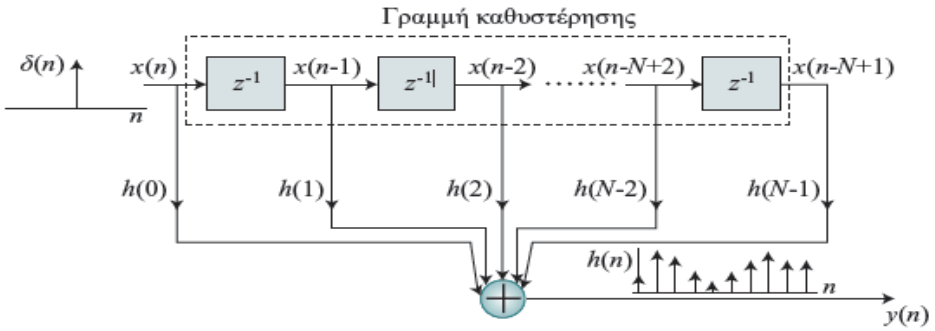
1. βαθυπερατό (low-pass)
2. υψηπερατό (high-pass)
3. διέλευσης ζώνης (band-pass)
4. αποκοπής ζώνης (band-stop)

Η δομή ενός FIR φίλτρου είναι από:

1. Από μία γραμμή καθυστέρησης (delay line) για ολίσθηση των δεδομένων x(n)
2. Ένα αθροιστή για την άθροιση όλων των γινομένων της εξόδου y(n)
3. Έναν τουλάχιστο πολλαπλασιαστή για τον υπολογισμό των σταθερών του φίλτρου με τα καθυστερημένα σήματα εισόδου b(k)

Σύμφωνα με το σχήμα 2.3 η δομή ενός FIR φίλτρου αποτελείται από μία γραμμή καθυστέρησης (delay line) όπου ολισθαίνουν τα δείγματα του σήματος εισόδου x(n), και από τους πολλαπλασιαστές b(k).

Τα αποτελέσματα των πολλαπλασιασμών προστίθενται για να δώσουν την τελική έξοδο του φίλτρου y(n).



Σχήμα 2.3

Η έξοδος y(n) είναι ο γραμμικός συνδυασμός των δειγμάτων εισόδου x(n) και είναι σύμφωνα με την εξίσωση:

$$(2.1)y(n) = \sum_{K=0}^{N-1} b[k]x[n-k] \quad \textbf{(2.4)}$$

Αν στην είσοδο του φίλτρου έχουμε μοναδιαία κρουστική απόκριση δ(n), η έξοδός του φίλτρου (η κρουστική απόκριση του φίλτρου ή Impulse Response) θα είναι διαδοχικά ίση με κάθε έναν από τους συντελεστές b(k).

Αυτό συμβαίνει γιατί η δ(n) θα πολλαπλασιάζεται κάθε φορά μόνο με έναν από τους συντελεστές b(k) καθώς ολισθαίνει στη γραμμή καθυστέρησης.

Η κρουστική απόκριση του φίλτρου θα μηδενιστεί όταν η μοναδιαία κρούση εξέλθει της γραμμής καθυστέρησης. Επομένως θα λάβουμε πεπερασμένη σε χρονική διάρκεια κρουστική απόκριση h(n).

Η εξίσωση 2.4 μπορεί να γραφεί ως εξής:

$$(2.3)y(n) = \sum_{K=0}^{N-1} h[k]x[n-k] \quad \textbf{(2.5)}$$

Μετασχηματίζοντας και τα δύο μέλη της τελευταίας σχέσης στο πεδίο -z θα λάβουμε τη συνάρτηση μεταφοράς του FIR φίλτρου:

$$y(z)=\sum_{k=0}^{N}h(k)x(z)z^{-k}\Rightarrow y(z)=y(z)\sum_{k=0}^{N}h(k)z^{-k}\Rightarrow H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(\quad \textbf{(2.6)}$$

Τα FIR φίλτρα είναι ευσταθή, γεγονός που οφείλεται στο ότι δεν έχουμε πόλους στη συνάρτηση μεταφοράς και οι μηδενισμοί εφόσον είναι μιγαδικοί θα πρέπει να είναι συζυγείς έτσι ώστε να έχουμε συναρτήσεις με πραγματικούς συντελεστές.

Τα φίλτρα αυτά παρουσιάζουν γραμμική απόκριση φάσης.

Για να έχει ένα FIR φίλτρο την ιδιότητα της γραμμικής απόκρισης φάσης θα πρέπει να παρουσιάζει κάποιας μορφής συμμετρία στην κρουστική απόκριση h(n). Υπάρχουν τέσσερις διαφορετικοί τύποι FIR φίλτρων γραμμικής φάσης, όπου ανάλογα με το αν το πλήθος N των συντελεστών της h(n) είναι άρτιο ή περιττό και αν η h(n) είναι συμμετρική ή αντισυμμετρική. Ακολουθεί ο πίνακας 2.2 που περιγράφει τους τύπους FIR φίλτρων.

ΤΥΠΟΥ I	h(n) = h(N-n) (συμμετρική)	N άρτιο
ΤΥΠΟΥ II	h(n) = h(N-n) (συμμετρική)	N περιττό
ΤΥΠΟΥ III	h(n) = -h(N-n) (αντισυμμετρική)	N άρτιο
ΤΥΠΟΥ IV	h(n) = -h(N-n) (αντισυμμετρική)	N περιττό

Πίνακας 2.2

2.3 Σχεδίαση FIR φίλτρων

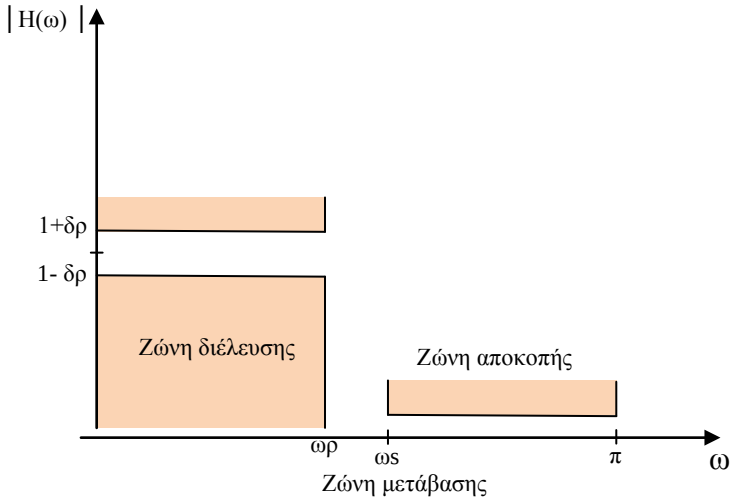
Η σχεδίαση ενός φίλτρου FIR περιλαμβάνει την αναζήτηση των συντελεστών h(n) που επιστρέφουν μια απόκριση συχνότητας η οποία ικανοποιεί τις προδιαγραφές που ήδη έχουν καθορισθεί για το σχεδιασμό του φίλτρου.

Τα βήματα που πρέπει να ακολουθήσουμε για το σχεδιασμό των ψηφιακών φίλτρων είναι:

α. Προσδιορισμός προδιαγραφών

- Συχνότητα αποκοπής ωp στη ζώνη διάβασης,
- Συχνότητα αποκοπής ωs στη ζώνη αποκοπής,
- Απόκλιση δp στη ζώνη διέλευσης και
- Απόκλιση δs στη ζώνη αποκοπής.
- Το διάστημα Δω= ωs - ωp καλείται ζώνη μετάβασης.
- Η συχνότητα αποκοπής ωc = (ωs + ωp) / 2 αντιστοιχεί στο μέσο της ζώνης μετάβασης.
- Η απόκλιση στις ζώνες διέλευσης και αποκοπής σε decibels (dB) υπολογίζονται σύμφωνα με τους εξής τύπους:
- αp = -20 log(1 – δp)

- αs = -20 log(δs)



Σχήμα 2.4

β.**Υπολογισμός των σταθερών των φίλτρων**

Μέθοδοι υπολογισμού h(n):

- Ισοκυματικά (Equiripple)
- Least-squares
- Μέθοδος παραθύρων(Window)
- Const.least-squares
- Complex equiripple
- Maximally flat
- Least pth-norm
- Constuined-equiripple
- Generalized-equiripple
- Const.band equiripple
- Intepluded FIR

2.4. **Σχεδίαση FIR φίλτρων με τη μέθοδο παραθύρων (window)**

2.4.1 Μέθοδος των παραθύρων

Η μέθοδος των παραθύρων βασίζετε στον αντίστροφο μετασχηματισμό Furier (IDTFT)

$$hd(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} h(e^{j\omega})e^{jn\omega}d\omega$$

Δίνεται η μορφή της απόκρισης συχνότητας Η(ω) και ζητείται η αντίστοιχη h(n).

Συνήθως εφαρμόζετε για απλές μορφές Η(ω). Το πρόβλημα της μεθόδου των

παραθύρων είναι ο αριθμός των συντελεστών h(n) που πρέπει να επιλεγούν. Επιλέγεται το παράθυρο από την επιθυμητή εξασθένηση στη ζώνη αποκοπής, σύμφωνα με τον ακόλουθο πίνακα.

Παράθυρο	Πλάτος πλευρικού λοβού	Εύρος ζώνης μετάβασης	Εξασθένιση στη ζώνη αποκοπής
Ορθογώνιο (rectangular)	-13	0.9/N	-21
Hamming	-31	3.1/N	-44
Hanning	-41	3.3/N	-53
Blackman	-57	5.5/N	-74

Πίνακας 2.3

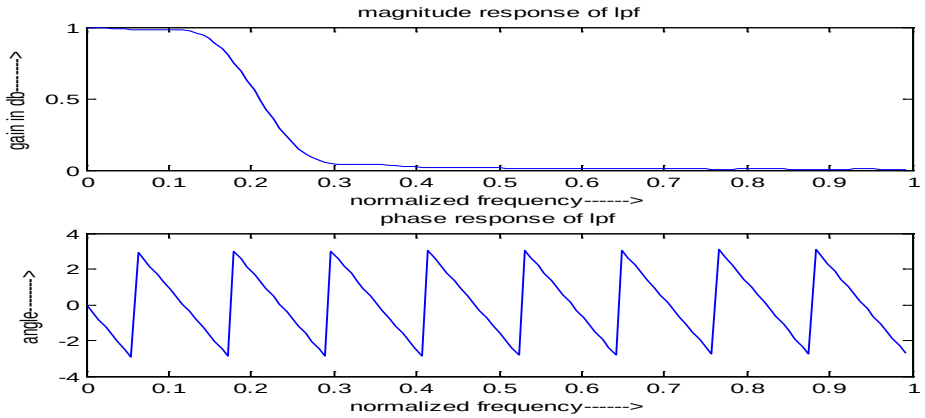
Ακολούθος βρίσκεται η τάξη του φίλτρου από το εύρος της ζώνης μετάβασης (όσο μεγαλύτερη η τάξη τόσο πιο στενή η ζώνη μετάβασης). Η τάξη υπολογίζεται από τον προηγούμενο πίνακα, ανάλογα με τον τύπο του παραθύρου και το επιθυμητό εύρος της ζώνης μετάβασης. Για παράδειγμα εάν έχουμε Hanning παράθυρο και εύρος ζώνης μετάβασης Δω, τότε ισχύει η σχέση Δω=3.1/N απο την οποία προκύπτει N=3.1/Δω όπου N η τάξη του φίλτρου. Στη συνέχεια βρίσκονται οι συντελεστές h(n) =sin(nωc)/nπ οπου n ανήκει στο διάστημα [1, N]. Οι τελικοί συντελεστές βγαίνουν από τη σχέση h(n)*w(n) οπου το w(n) ειναι το παράθυρο και εξαρτάται από τον τύπο του παραθύρου, όπως παρακάτω:

Παράθυρο	$0 \le n \le N-1$
Ορθογώνιο (rectangular)	$w(n) = 1$
Hamming	$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right)$
Hanning	$w(n) = 0.5 - 0.5\cos\left(\frac{2\pi n}{N-1}\right)$
Blackman	$w(n) = 0.42 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) + 0.8\cos\left(\frac{4\pi\pi}{N-1}\right)$

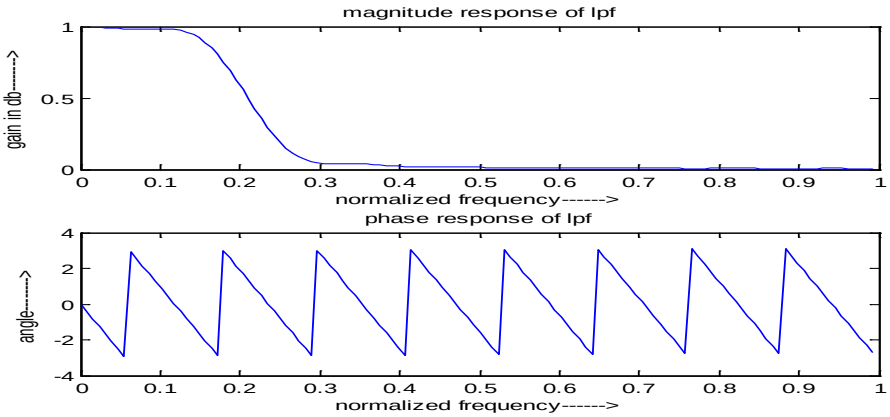
Πίνακας 2.4

2.4.2 Όλα τα Παράθυρα

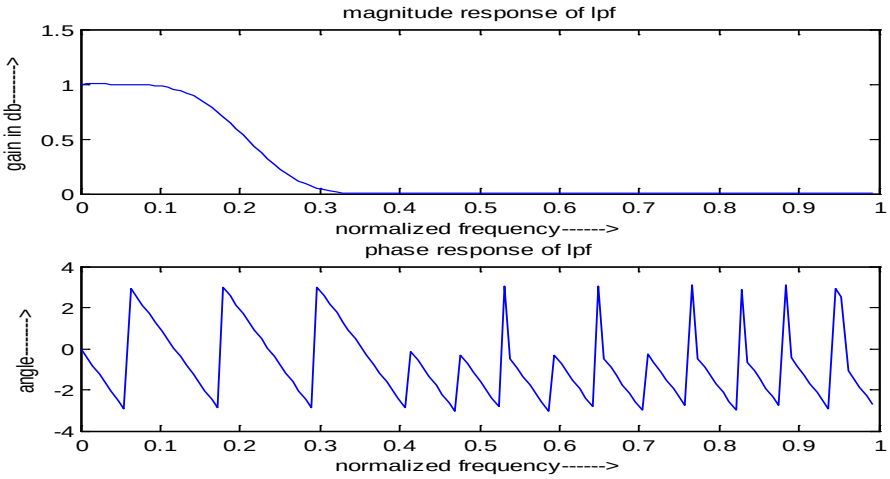
1. w = barthannwin(N)



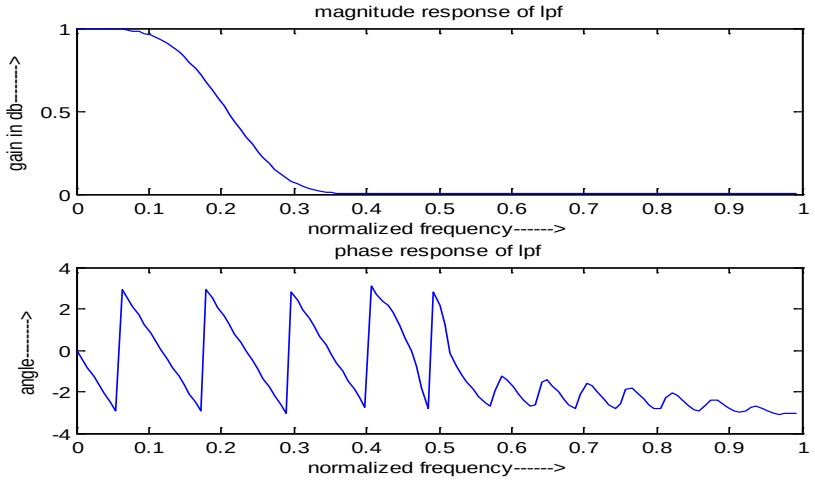
2. w = bartlett(N)



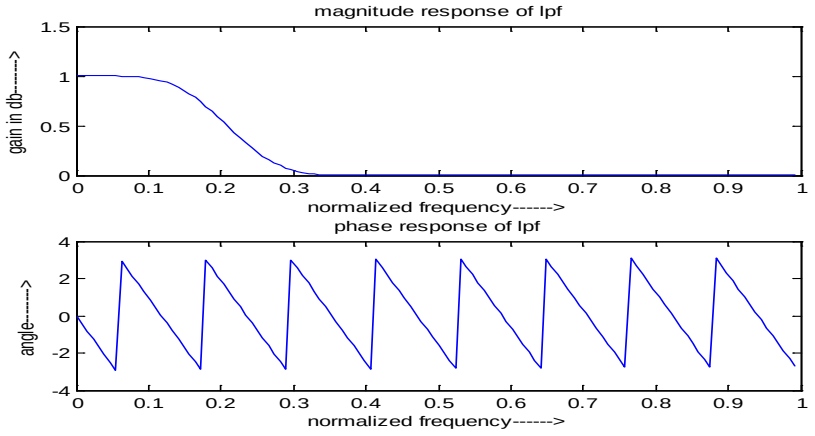
3. w = blackman(N)



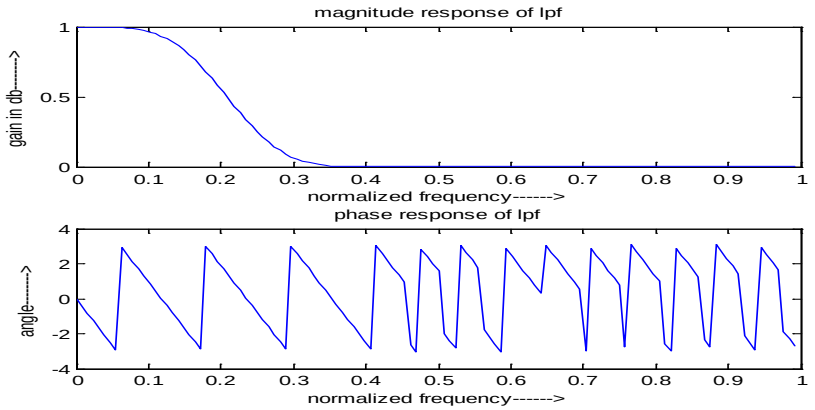
4. w = blackmanharris(N)



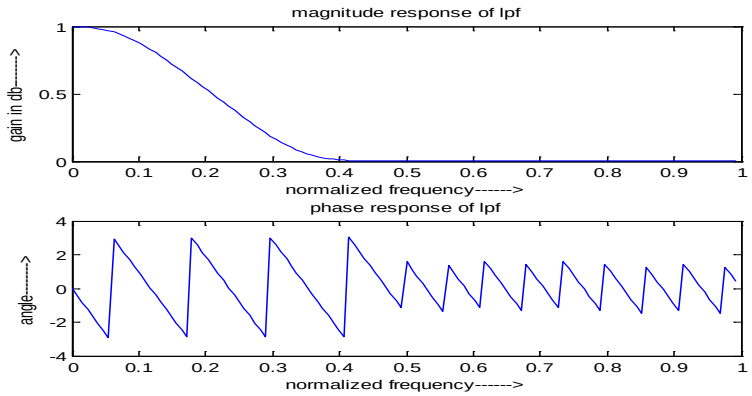
5. w = bohmanwin(N)



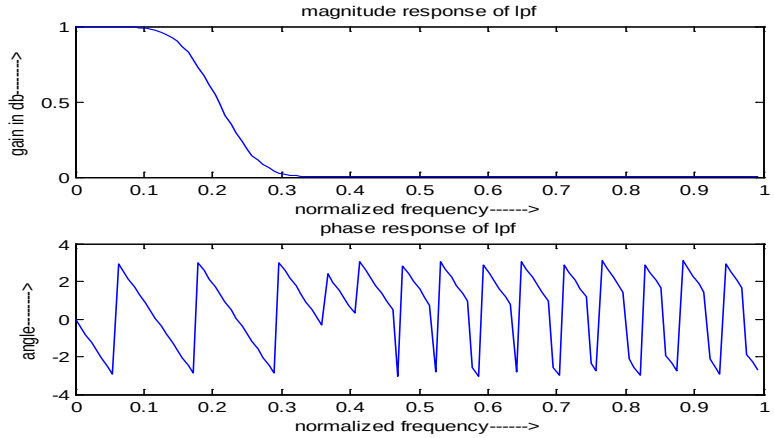
6. w = chebwin(N)



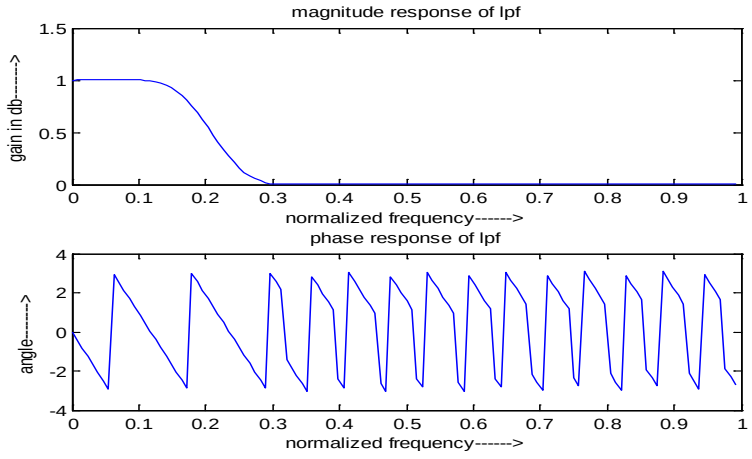
7. w = flattopwin(N)



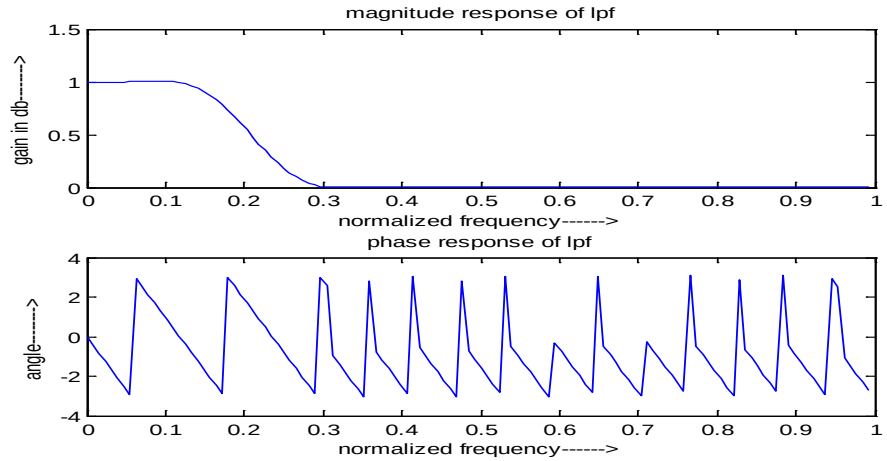
8. w = gausswin(N)



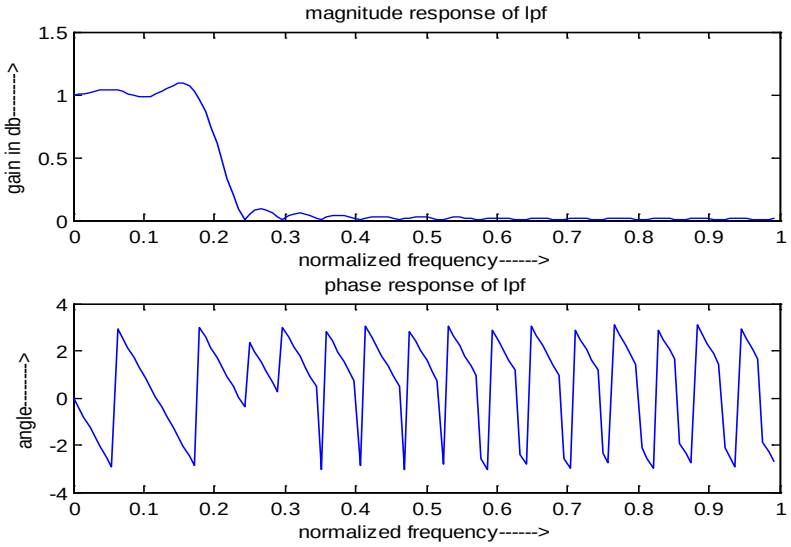
9. w = hamming(N)



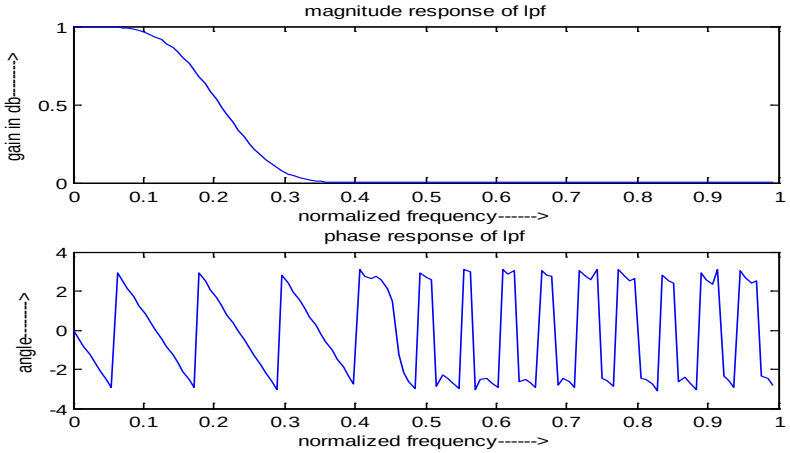
10. w = hann(N)



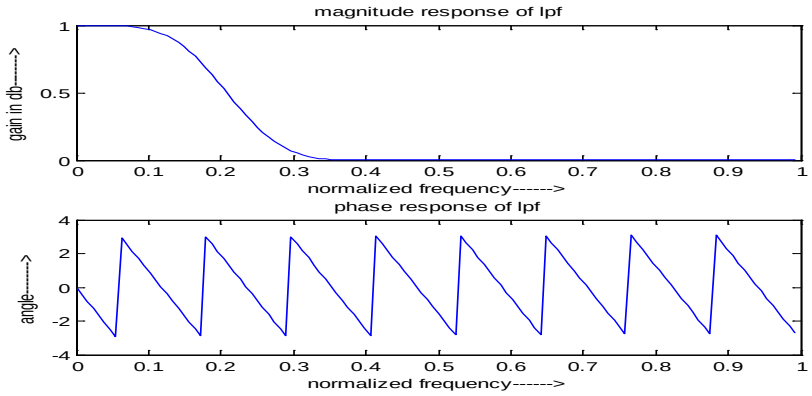
11. w = kaiser(N)



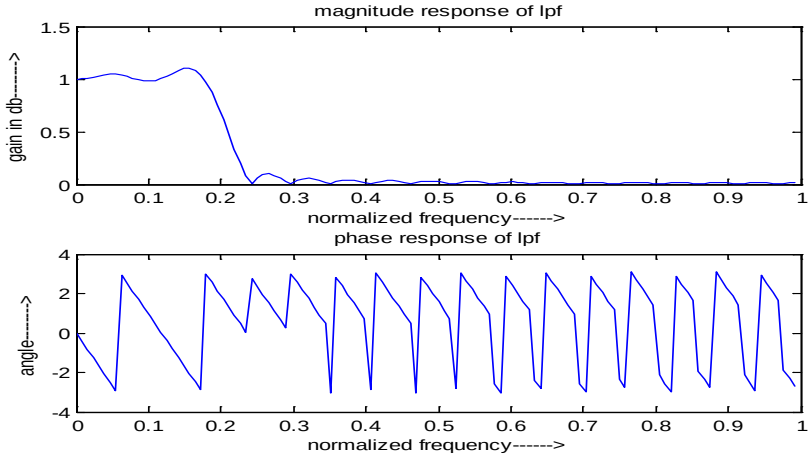
12. w = nuttallwin(N)



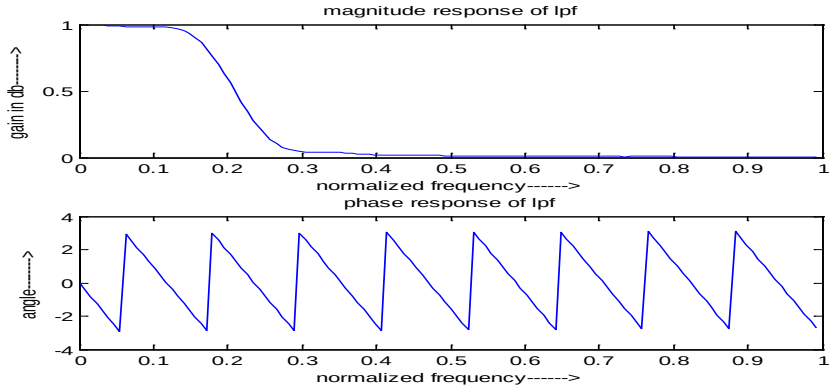
13. w = parzenwin(N)



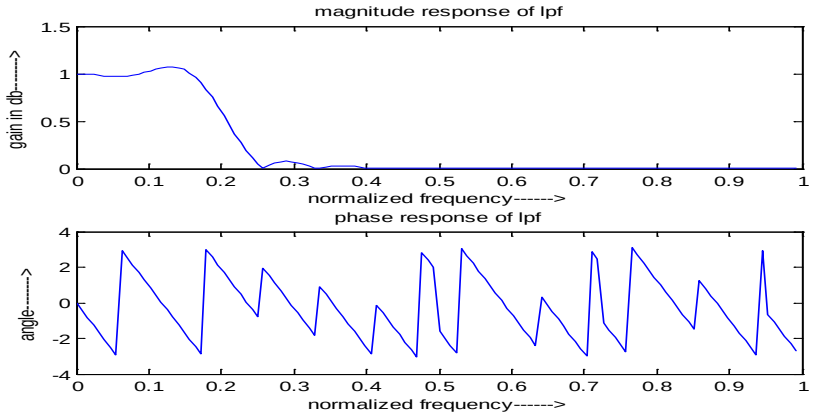
14. w = rectwin(N)



15. w= triang(N)

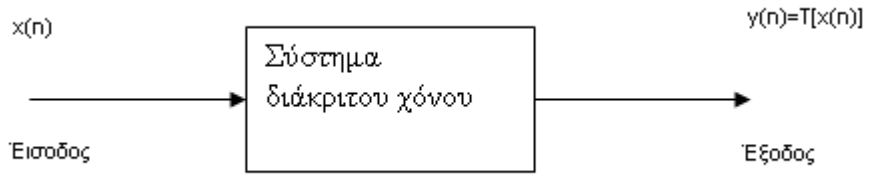


16. w = tukeywin(N)



2.5 Ψηφιακά φίλτρα άπειρης κρουστικής απόκρισης IIR (infinite impulse response)

Ένα σύστημα διάκριτου χρόνου είναι μια υπολογιστική διαδικασία που δέχεται στην είσοδο μια σειρα αριθμών x(n)και τα μετασχηματίζει σε μια σειρά αριθμών y(n)που είναι η έξοδος .



Η λειτουργεια του βασίζεται απο την σχέση y(n)=T[x(n)] η οποία δίνει την έξοδο, όταν μια συνάρτηση T[ ] εφαρμόζετε στα δείγματα εισόδου x[n].

Καθώς τα συστήματα διάκριτου χρονου είναι μια σειρά από αριθμούς, αυτές οι διεργασίες περιγράφονται όταν γνωρίζουμε την συνάρτηση T[ ].

Η γενική σχέση που περιγράφει ένα IIR φίλτρο φαίνεται στην εξίσωση(2.5)

$$(2.5)\, y(n) = \sum_{k=0}^{\infty} h(k)x(n-k) = \sum_{k=0}^{N} b(k)x(n-k) - \sum_{K=1}^{M} a(k)y(n-k)$$

όπου h(k)η κρουστική απόκριση του φίλτρου ,x(n) η ακολουθία εισόδου, y(n) η ακολουθία εξόδου και b(k), a(k) οι συντελεστές του φίλτρου. Η εξίσωση αυτή μας λέει ότι η τρέχουσα τιμή του δείγματος εξόδου προκύπτει ως γραμμικός συνδυασμός του παρόντος και προηγουμένων δειγμάτων της εισόδου, καθώς και από προηγούμενες τιμές της εξόδου.
Η συνάρτηση μεταφοράς των IIR φίλτρων H(z) δίνεται απο τη σχέση (2.6)

$$(2.6) H(z) = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{1 + \sum_{k=1}^{M} a(k) z^{-k}}$$

2.5.1 Κρουστική απόκριση h[n]

Όταν η είσοδος σε ένα IIR φίλτρο είναι η μοναδιαια κρουστική ακολουθία **X(n)=δ(n),** η έξοδος ονομάζεται κρουστική απόκριση άπειρου μήκους και συμβολίζετε με h[n].
Εφαρμογή (2.5.1)
y(n) = 0.5y(n-1) + x(n)
Αντικαθιστώντας x(n)=δ(n) στην εξίσωση διαφοράς

$$(2.5) y(n) = \sum_{k=0}^{\infty} h(k) x(n-k) = \sum_{k=0}^{N} b(k) x(n-k) - \sum_{K=1}^{M} a(k) y(n-k)$$

y(n) = 0.2 y(n-1) + x(n)
y0 = x0 + y-1
y1 = x1 + y0
y2 = x2 + y1
y3 = x3 + y2
προκύπτει η ζητούμενη κρουστική απόκριση (έξοδος συστήματος)
h(0)=0.2 * 0 + 1 = 1,
h(1)=0.2 * 1 + 0 = 0.2,
h(2)=0.2 * 0.2 + 0 =0.04
h(3)=0.2 * 0.04 + 0 = 0.008
και γενικά h(n) = 0.2n n≥0
Βλέπουμε άπειρη σε μήκος κρουστική απόκριση (αν και οι τιμές μειώνονται γρήγορα όσο το n αυξάνει) όταν η εξίσωση διαφοράς περιέχει στο δεξιό μέρος όχι μόνο προηγούμενα δείγματα εισόδου αλλά και προηγούμενα δείγματα εξόδου.

2.6 Χαρακτηριστικά και είδη IIR φίλτρων

Τα IIR φίλτρα, όπως προαναφέρθηκε, είναι τα ψηφίακα φίλτρα που έχουν άπειρη χρονική διάρκεια όταν στην είσοδο τους εφαρμοστεί μοναδιαία κρουστική απόκριση δ(n).

Υπάρχουν τέσσερα βασικά είδη IIR φίλτρων :

1. βαθυπερατό (low-pass)
2. υψηπερατό (high-pass)
3. διέλευσης ζώνης (band-pass)
4. αποκοπής ζώνης (band-stop)

Η έξοδος y(n) εξαρτάται και από προηγούμενες εξόδους. Δηλαδή η εξίσωση διαφορών (2.5)θα έχει την εξής μορφή (2.6)

α0y(n)+α1y(n-1)+. . . +αNy(n-N)=b0x(n)+b1x(n-1)+. . .+bMx(n-M) (2.6)

Η συνάρτηση μεταφοράς θα έχει την εξής μορφή :

$$(2.6)H(z)=\frac{b0+b1z^{-1}+\ldots+bNz^{-N}}{1+a1z^{-1}+\ldots+aMz^{-M}}=\frac{\sum_{k=0}^{N}b(k)z^{-k}}{1+\sum_{k=1}^{M}a(k)z^{-k}}=\frac{k(z-z1)(z-z2)\ldots(z-zN)}{(z-p1)\ldots(z-pM)}$$

Από την εξίσωση (2.6)βλέπουμε οτι έχουμε μη μηδενικούς πόλους.

Όπου zi= ρίζες,pi=πόλοι

Χαρακτηριστικά IIR φίλτρων.

- Απαιτούν μικρό αριθμό συντελεστών
- Δεν έχουν γραμμική φάση.
- Η υλοποίηση της συνέλιξης που περιγράφουν τα IIR φίλτρα απαιτεί ψηφιακό κύκλωμα με ανατροφοδότηση.
- Η έξοδος ενός IIR φίλτρου υπολογίζεται χρησιμοποιώντας εκτός από τις τιμές της εισόδου και τις προηγούμενες τιμές της εξόδου.
- Ενώ τα FIR υλοποιούνται χωρίς ανατροφοδότηση με αποτέλεσμα να είναι πάντα ευσταθή και να επηρεάζονται ελάχιστα από σφάλματα κβάντισης του σήματος ή των συντελεστών του φίλτρου.
- Η κυμάτωση στην ζώνη διέλευσης για τα IIR φίλτρα θεωρείται η από κορυφή σε κορυφή απόκλιση.
- Όταν ένας μηδενισμός τοποθετείται σε ένα σημείο του επιπέδου z, η απόκριση συχνότητας στο αντίστοιχο σημείο είναι μηδέν.
- Αντίθετα ένας πόλος δημιουργεί μια αιχμή στην αντίστοιχη συχνότητα.
- Πάνω στον μοναδιαίο κύκλο οι συχνότητες μεταβάλλονται από το 0 έως το Fs με το 0 στο δεξιό μέρος του κύκλου ή από -Fs/2 έως Fs/2.

- Πόλοι κοντά στον μοναδιαίο κύκλο δίνουν αιχμηρά μέγιστα στην απόκριση συχνότητας, ενώ μηδενισμοί κοντά ή πάνω στον μοναδιαίο κύκλο δίνουν ελάχιστα στην απόκριση συχνότητας.
- Να σημειωθεί πως οι συντελεστές του φίλτρου θα πρέπει να είναι πραγματικοί ή να εμφανίζονται σε ζεύγη συζυγών μιγαδικών.
- Επιπλέον για να είναι ευσταθή τα IIR φίλτρα πρέπει έχουν όλους τους πόλους μέσα στον μοναδιαίο κύκλο.

2.7 Σχεδίαση IIR φίλτρων

Τα IIR φίλτρα μπορούν να σχεδιαστούν άμεσα στο πεδίο του μετασχηματισμού-z αλλά συνήθως ο σχεδιασμός τους βασίζεται στα αντίστοιχα αναλογικά που σχεδιάζονται στο επίπεδο- s. Με μετασχηματισμό s→z λαμβάνεται το ψηφιακό φίλτρο από το αντίστοιχο αναλογικό φίλτρο. Τέτοιοι μετασχηματισμοί είναι ο διγραμμικός και ο μετασχηματισμός αμετάβλητης κρουστικής απόκρισης.

2.7.1 Σχεδίαση IIR φίλτρου απο αναλογικό σε ψηφιακο

Τα βασικά είδη συναρτήσεων που χρησιμοποιούνται στο σχεδιασμό αναλογικών φίλτρων είναι

1. Butterworth,
2. ChebyshevI
3. ChebyshevII
4. Elliptic

1.Φίλτρα Butterworth

Ένα βαθυπερατό φίλτρο Butterworth είναι ένα φίλτρο μόνο με πόλους .

Το πλάτος της απόκρισης συχνότητας δίνεται απο τη σχέση (2.7)

$$\left|H(j\omega)\right|^2 = \frac{1}{1+(\frac{J\omega}{J\omega c})^{2N}} (2.7)$$

Όπου N είναι η τάξη του φίλτρου (προσδιορίζεται απο τον αριθμό των πόλων της συνάρτησης μεταφοράς του φίλτρου)και ωc είναι η συχνότητα αποκοπής στα 3Db.

Όσο αυξάνει η συχνότητα αποκοπής ωc,η απόκριση συχνότητας του φίλτρου φθίνει μονότονα.Η αύξηση της τάξης N του φίλτρου μειώνει τη ζώνη μετάβασης .

2.Φίλτρα Chebyshev τύπου I

Ένα βαθυπερατό φίλτρο Chebyshev τυπου I είναι επίσης ένα φίλτρο μόνο με πόλους .

Το πλάτος της απόκρισης συχνότητας δίνεται απο τη σχέση (2.7.2)

$$\left|H(J\omega)\right|^2 = \frac{1}{1+\varepsilon^2 \mathrm{T}^2 \mathrm{N}(\frac{\omega}{\omega p})} (2.7.2)$$

Όπου N η τάξη του φίλτρου,ωp η συχνότητα αποκοπής στη ζώνη διελευσης ,TN το πολυώνυμο Chebyshev και ε μια παράμετρος που ελέγχει τη κυμάτωση στη ζώνη διέλευσης .Τα φίλτρα Chebyshev I παρουσιάζουν ομοιόμορφη κυμάτωση στη ζώνη διέλευσης και μονότονη φθίνουσα ζώνη αποκοπής .Η αύξηση της ταξης N του φίλτρου αυξάνει τον αριθμό των ταλαντώσεων στη ζώνη διέλευσης και μειώνει το εύρος της ζώνης μετάβασης .

3.Φίλτρα Chebyshev τύπου II

Τα φίλτρα chebyshev II έχουν και πόλους και μηδενικά .Το πλάτος της απόκρισης συχνότητας δίνεται απο τη σχέση (2.7.3)

$$\left|H(j\omega)\right|^2 = \frac{1}{1+\varepsilon^2\left[\frac{\mathrm{TN}(\frac{\omega c}{\omega p})}{\mathrm{TN}(\frac{\omega s}{\omega})}\right]^2} \quad (2.7.3)$$

Όπου N η τάξη του φίλτρου ,ωp και ωs οι συχνότητες αποκοπής στη ζώνη διέλευσής και στη ζώνη αποκοπής ,TN το πολυώνυμο chebyshev και ε μια παράμετρος που ελέγχει τη κυμάτωση στη ζώνη αποκοπής.

Τα φίλτρα chebyshev II παρουσιάζουν μονοτονική συμπεριφορά στη ζώνη διέλευσης και ισοκυματική ζώνη αποκοπής .Η αύξηση της τάξης του φίλτρου μειώνει το εύρος της ζώνης μετάβασης και αυξάνει τον αριθμό των ταλαντώσεων.

4.Φίλτρα Elliptic

Τα ελλειπτικά φίλτρα παρουσιάζουν ομοιόμορφη κυμάτωση τόσο στη ζώνη διέλευσης, όσο και στη ζώνη αποκοπής.Τα φίλτρα αυτά έχουν πόλους και μηδενικά και το μέτρο της απόκρισης στη συχνότητα δίνεται από τη σχέση(2.7.4)

$$\left|H(J\omega)\right|^2 = \frac{1}{[1+\varepsilon^2 U\mathrm{N}(\frac{\omega}{\omega p})]^{1/2}} \quad (2.7.4)$$

2.7.2 Διγραμμικός Μετασχηματισμός z (ΔΜΖ)

Έχοντας δεδομένη την συνάρτηση μεταφοράς H(s) του αναλογικού φίλτρου, η ζητούμενη συνάρτηση μεταφοράς H(z) του ψηφιακού φίλτρου προκύπτει μέσω συγκεκριμένων μεθόδων. Μία από αυτές τις μεθόδους είναι ο διγραμμικός μετασχηματισμός. Η σχέση μετασχηματισμού είναι η ακόλουθη:

$$s = k\frac{z-1}{z+1} \quad k = 1\acute{\eta}\frac{2}{\mathrm{T}}$$

Βαθυπερατό ήΥψηπερατό φίλτρο: συχνότητα αποκοπής ωc

$$\omega c = k\tan\left(\frac{\omega cT}{2}\right)$$

$$k = 1\acute{\eta}\frac{2}{\mathrm{T}}$$

Διέλευσης ζώνης ή Αποκοπής ζώνης ωc1και ωc2

$$\omega c1 = \tan\left(\frac{\omega c1T}{2}\right), \omega c2 = \tan\left(\frac{\omega c2T}{2}\right)$$

Αντικατάσταση s στο H(s) με :

Βαθυπερατό σε Βαθυπερατό : $s = \frac{s}{\omega c}$

Βαθυπερατό σε Υψηπερατό : $s = \frac{\omega c}{s}$

Βαθυπερατό σε Διέλευσης : $s = \frac{s^2 + \omega 0^2}{Ws}$

Βαθυπερατό σε Αποκοπής : $s = \frac{Ws}{s^2 + \omega 0^2}$

$$\omega 0^2 = \omega c1.\omega c2 \qquad W = \omega c2 - \omega c1$$

2.8 Διαφορές FIR και IIR

Χαρακτηριστικά FIR

1. Τα FIR φίλτρα έχουν γραμμική φάση με αποτέλεσμα να μην παραμορφώνουν τα σήματα για τα οποία η φάση είναι καθοριστική.
2. Τα FIR υλοποιούνται χωρίς ανατροφοδότηση με αποτέλεσμα να είναι ευσταθή και να επηρεάζονται ελάχιστα από σφάλματα κβάντισης του σήματος ή των συντελεστών του φίλτρου.
3. Η χρήση πεπερασμένου αριθμού bits στην αναπαράσταση των συντελεστών έχει ως αποτέλεσμα την μείωση της απόδοσης των FIR φίλτρων.

4. Κάθε αύξηση κατά 1 bit στην ακρίβεια των συντελεστών βελτιώνει την απόδοση του φίλτρου κατά 6dBs.
5. Δεν είναι δυνατό ένα αναλογικό φίλτρο να μετατραπεί σε ψηφιακό στα FIR φίλτρα.

Χαρακτηριστικά IIR

1. Καταρχήν τα IIR φίλτρα έχουν μη-πεπερασμένη κρουστική απόκριση.
2. Η υλοποίηση της συνέλιξης που περιγράφουν τα IIR φίλτρα απαιτεί ψηφιακό κύκλωμα με ανατροφοδότηση.
3. Για τον υπολογισμό της εξόδου εκτός από τις τιμές της εισόδου χρησιμοποιούνται και οι προηγούμενες τιμές της εξόδου.
4. Για την υλοποίηση ενός IIR φίλτρου απαιτούνται λιγότεροι συντελεστές σε σχέση με αυτούς που απαιτούνται για την υλοποίηση του αντίστοιχου FIR φίλτρου .
5. Η επιλογή γενικά ανάμεσα στα δύο είδη φίλτρων, FIR και IIR, εξαρτάται τόσο από την εφαρμογή όσο και από τα ιδιαίτερα χαρακτηριστικά των φίλτρων.

ΚΕΦΑΛΑΙΟ 3

TMS320C6713 DSK και το περιβάλλον του CCS

3.TMS320C6713 DSK και το περιβάλλον του Code Composer Studio.

Ένα σύστημα ψηφιακής επεξεργασίας σήματος (Digital Signal Processing System) μπορεί να οριστεί σαν ένα ηλεκτρονικό σύστημα το οποίο κάνει χρήση ψηφιακής επεξεργασίας σήματος. Όπως είναι γνωστό τα σήματα αναπαρίστανται ψηφιακά σαν ακολουθία από δείγματα. Συνήθως αυτά προέρχονται από φυσικά σήματα π.χ. ηχητικά σήματα) μέσω της χρήσης ενός μετατροπέα (όπως είναι τα μικρόφωνα) καθώς και αναλογικών / ψηφιακών μετατροπέων (A/D converters).

Μετά την επεξεργασία τους, τα ψηφιακά σήματα μετατρέπονται ξανά σε αναλογικά με τη χρήση ψηφιακών / αναλογικών μετατροπέων (D/A converters).

Παράδειγμα ενός ολοκληρωμένου περιβάλλοντος ανάπτυξης (Integrated Development Environment) που ενσωματώνει, εύκολα στην χρήση, software εργαλεία αποτελεί το Code Composer Studio (CCS) με την βοήθεια του οποίου γίνεται μελέτη και χρήση των DSPs της TI (Texas Instruments).

Μέσα στο ολοκληρωμένο περιβάλλον ανάπτυξης Code Composer Studio περιλαμβάνονται οδηγίες χρήσης για το πώς γίνεται εγκατάσταση του κατάλληλου εξομοιωτή, βιβλιοθήκες, αλλά και ο τρόπος με τον οποίο μπορεί κάποιος να δημιουργήσει ένα project, να το «φορτώσει», να το «τρέξει» στον κατάλληλο εξομοιωτή ή στην κάρτα, να δει τα αποτελέσματα σε γράφημα κτλ.

Το CCS περιλαμβάνει εργαλεία όπως ο C compiler, ο assembler και ο linker. Έχει γραφικές δυνατότητες και υποστηρίζει real-time debugging. Μπορεί κάποιος να γράψει κώδικα σε γλώσσα C, σε γλώσσα assembly, σε linear assembly ή και σε συνδυασμό αυτών.

3.1Εισαγωγή στη κάρτα dsk c6713

Η πλατφόρμα dsk c6713 σχεδιάστηκε και κατασκευάστηκε από την(Texas Instruments) και θεωρείται μετεξέλιξη του dsk c6711 ενσωματώνοντας επιπλέον χαρακτηριστικά και τεχνολογίες χωρίς να χάνει την λειτουργικότητα του και την ευχρηστία του.

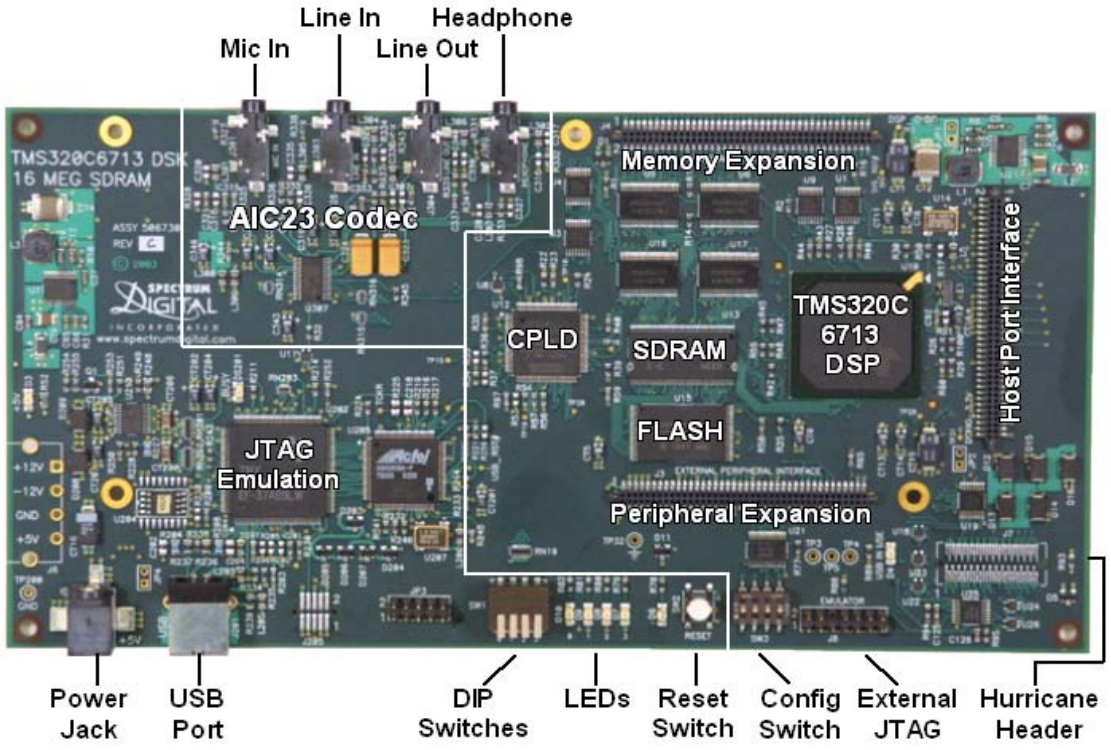
Η κάρτα dsk c6713 είναι μια χαμηλού κόστους πλατφόρμα που αφήνει τους χρήστες να αναπτύξουν εφαρμογές για την οικογένεια c6713 DSP.

3.2 Δομή της κάρτας dsk c6713

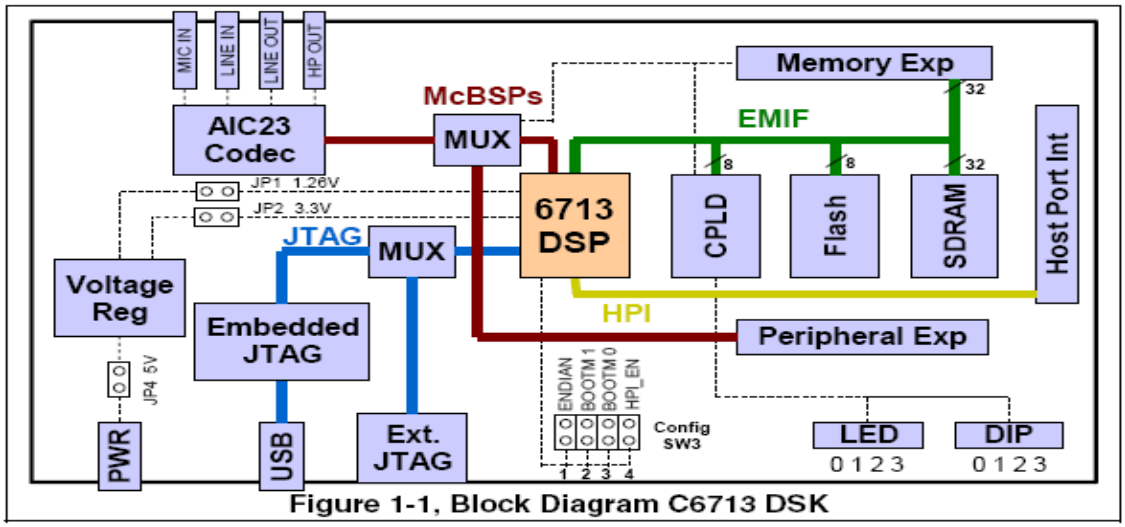
Η κάρτα dsk c6713 αποτελείται από τα πιο κάτω τμήματα:

- Τον DSP TMS320C6713 που λειτουργεί στα 225 MHz.
- Ένα AIC23-32-bit stereo codec.
- Τέσσερα Software accessible LEDs και DIP Switches που δίνουν στο χρήστη μια πολύ απλή μορφή εισόδου / εξόδου.
- Μια 512Mbyte flash μνήμη της κάρτας.
- Μια 16MByte Synchronous DRAM.
- JTAG emulation μέσω του on-board JTAG emulator, ο οποίος είναι προσιτός μέσω USB.
- Ρυθμιστές τάσης.
- Το daughter card interface το οποίο παρέχει τρεις expansion connectors που μπορούν να δεχτούν διάφορες daughter cards ανάλογα με την εφαρμογή.
- Το cpld epm3128tc100-10 της altera, για την υλοποίηση τεσσάρων memory-mapped καταχωρητών που δίνουν την δυνατότητα για software έλεγχο διάφορων καταχωρητών.

Πιο κάτω φαίνονται τα κύρια τμήματα της κάρτας dskc6713 σε πραγματική μορφή (σχήμα 3.1), καθώς και διάγραμμα τους σε σχηματική μορφή(σχήμα 3.2).



Σχήμα 3.1



Σχήμα 3.2

3.3 Λειτουργία της dskc6713

Ο DSP TMS320C6713 είναι η καρδιά του συστήματος, λειτουργεί στα 225MHz και μπορεί να εκτελέσει μέχρι 1800 εκατομμύρια κινητής υποδιαστολής εντολές ανά δευτερόλεπτο.

Το CPLD EPM 3128TC 100-10 της ALTERA, χρησιμοποιείται για την υλοποίηση τεσσάρων memory-mapped καταχωρητών που δίνουν την δυνατότητα για software έλεγχο διάφορων καταχωρητών της κάρτας. Οι καταχωρητές είναι:

α. Ο καταχωρητής USER_REG διαβάζει την κατάσταση των τεσσάρων DIP-Switches και αναλόγως αναβοσβήνει τα LED.

β. Ο καταχωρητής DC_REG παρακολουθεί το Daughter Card Interface.

γ. Ο καταχωρητής Vercion περιέχει τις εκδόσεις του CPLD και BOARD.

δ. Ο καταχωρητής MISC ελέγχει το πώς τα δευτερεύοντα σήματα ελέγχονται από τους daughter card connectors.

Τα τέσσερα Software accessible LEDs και DIP Switches δίνουν στο χρήστη μια πολύ απλή μορφή εισόδου / εξόδου σύμφωνα με τον ακόλουθο πίνακα:

Switch 1	Switch 2	Switch 3	Switch 4	Configuration Description
Off				Little endian (default)
On				Big endian
	Off	Off		EMIF boot from 8-bit Flash (default)
	Off	On		HPI/Emulation boot
	On	Off		32-bit EMIF boot
	On	On		16-bit EMIF boot
			Off	HPI enabled on HPI pins (default)
			On	McASP1 enabled on HPI pins

Πίνακας 3.1

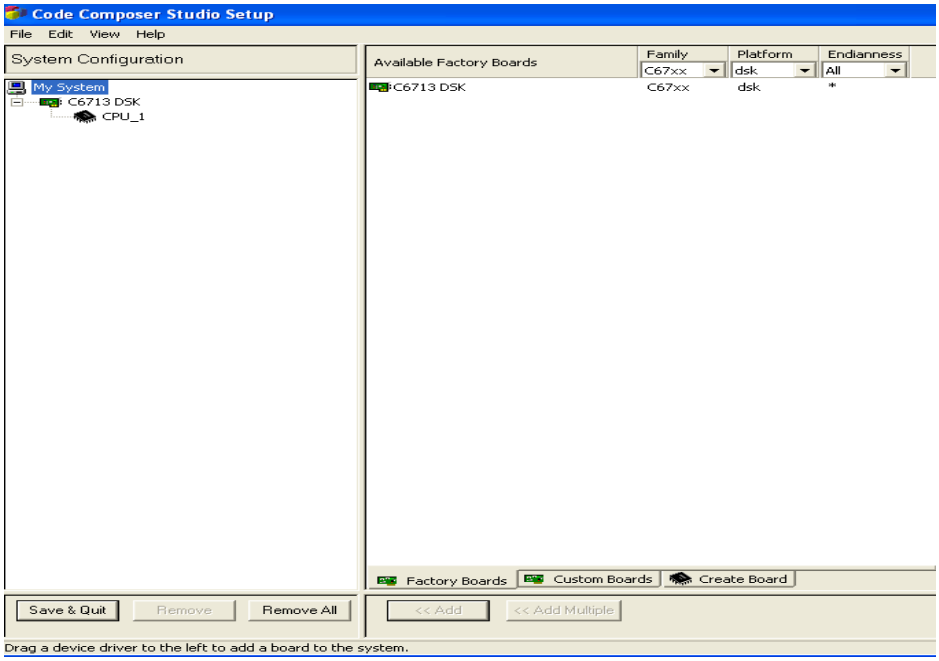
Η κάρτα έχει ένα AIC23-32-bit stereo codec το οποίο μετατρέπει το εισερχόμενο αναλογικό σήμα που έρχεται από τις εισόδους (MIC IN)και (LINE IN) σε ψηφιακό για να μπορεί να επεξεργαστεί από τον DSP και στην συνέχεια να ξαναμετατραπεί σε αναλογικό και να οδηγηθεί στις εξόδους HP OUT και LINE OUT.
Επίσης, η daughter card interface παρέχει τρεις expansion connectors που μπορούν να δεχτούν διάφορες daughter cards ανάλογα με την εφαρμογή, όπως πιο κάτω:

- Ο memory connector προσφέρει πρόσβαση στα ασύγχρονα EMIF σήματα του DSP για διασύνδεση τους με μνήμες και memory mapped συσκευές .
- Ο peripheral connector διαθέτει προς τα έξω τα διάφορα περιφερειακά σήματα του DSPόπως mcbsps , timers και clocks .
- Ο HPI connector είναι ένας υψηλής ταχύτητας interface που δίνει την δυνατότητα σε πολλούς dsp να συνεργάζονται μεταξύ τους.
-

3.4 Οδηγός για δημιουργία νέου project με την κάρτα c6713dsk

Βήμα 1

- Εγκατάσταση code composer studio 3.1 στον υπολογιστή.
- Σύνδεση κάρτας dsk6713 και ρύθμιση CCS 3.1 στον υπολογιστή.
- Έναρξη προγράμματος code composer studio 3.1 - setup code composer 3.1.
- Για να προστεθεί η κάρτα dsk6713 επιλέγουμε από το πεδίο family c67xx από το πεδίο platform επιλέγουμε dsk και από το πεδίο endianness επιλέγουμε all.
- Επιλέγουμε c6713DSK πατάμε add και μετά save and quit.
- Το πρόγραμμα είναι έτοιμο να χρησιμοποιηθεί.

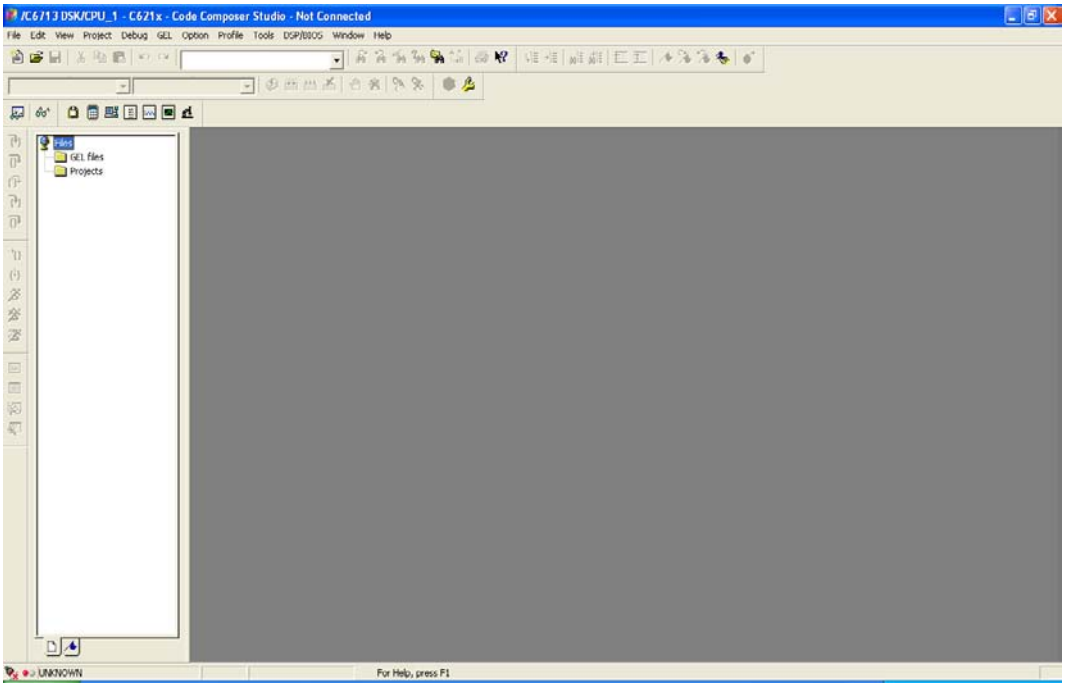


Σχήμα 3.3 (code composer studio setup)

Βήμα 2

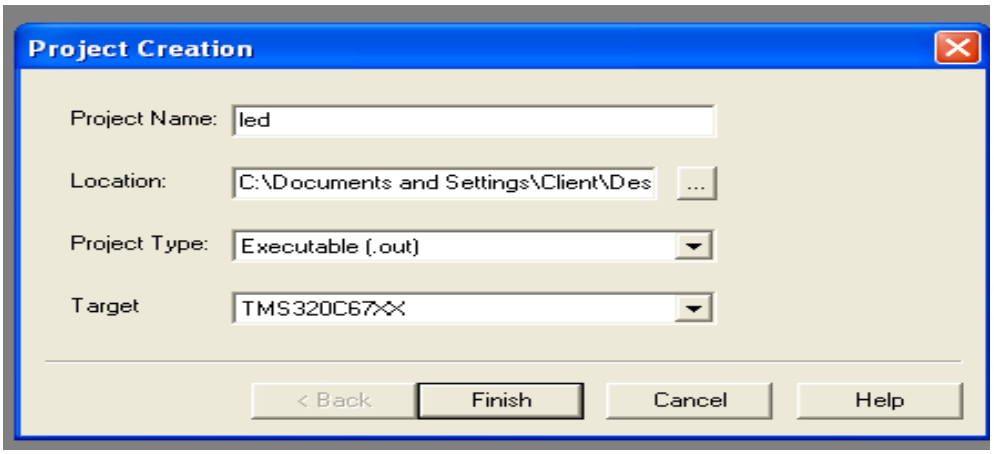
Δημιουργία new project στο CCS3.1.

- Έναρξη προγράμματα code composer studio 3.1. Εμφανίζετε το παράθυρο που φαίνεται στο σχήμα 3.4**.**



Σχήμα 3.4 (αρχικό παράθυρο CCS 3.1)

- Από το menu επιλέγουμε project-new και μας εμφανίζει το παράθυρο στο σχήμα3



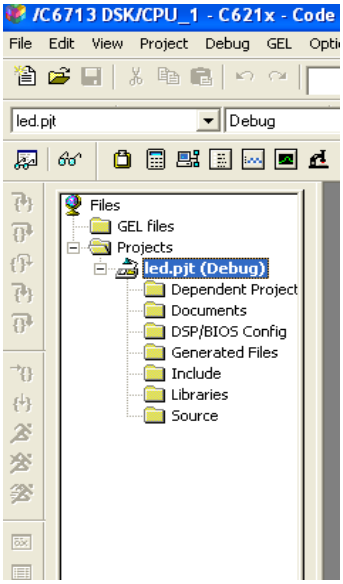
Σχήμα 3.5 (project creation)

- Στο project name εισάγουμε το όνομα του έργου μας .
- Στο location επιλέγουμε που θέλουμε να αποθηκεύσουμε το project μας . (C:\code composer studio 3.1 \ my project\led.c)
- Στο project type επιλέγουμε τον τύπο του project.executable(out).
- Στο target επιλέγουμε σε πια οικογένεια ανήκει το CHIP . TMS320C67XX.
- Finish.

Βήμα 3

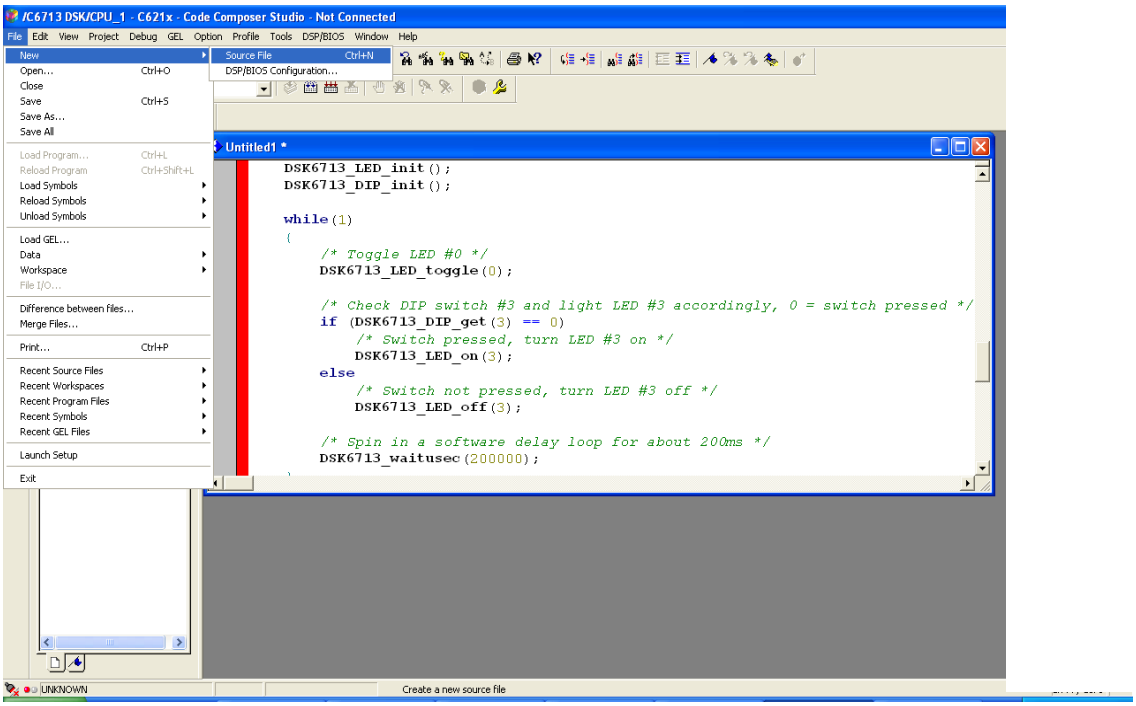
Κτίσιμο του project.

- Μετά την δημιουργία του project το CCS δημιουργεί ένα αρχείο προγράμματος led.pjt με τους υποφακέλλους του . Η μορφή του φαίνεται στο σχήμα 3.6.



Σχήμα 3.6 (αρχική μορφή project led.pjt)

- Εισάγουμε το πρόγραμμα μας που είναι γραμμένο σε γλώσσα C στο project led.pjt . Επιλέγουμε file –new –source file (σχήμα 3.7) όπου εμφανίζεται ένα παράθυρο όπου θα γράψουμε το κώδικα του προγράμματος μας και θα το αποθηκεύσουμε ως led.c στο (C:\code composer studio 3.1 \ my project\ \led.c).

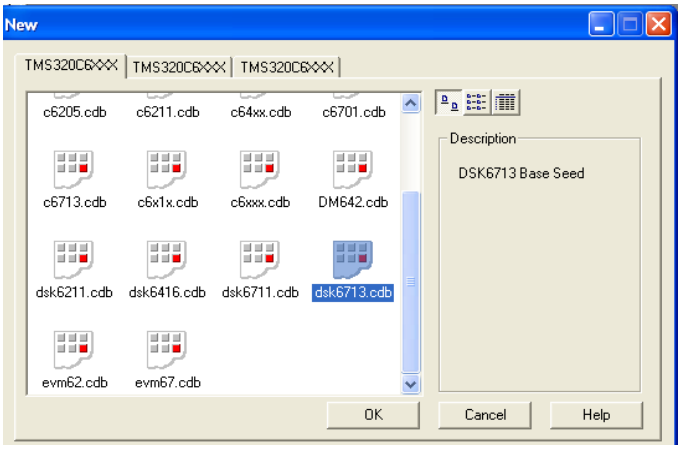


Σχήμα 3.7 (εισάγουμε το πρόγραμμα στο project)

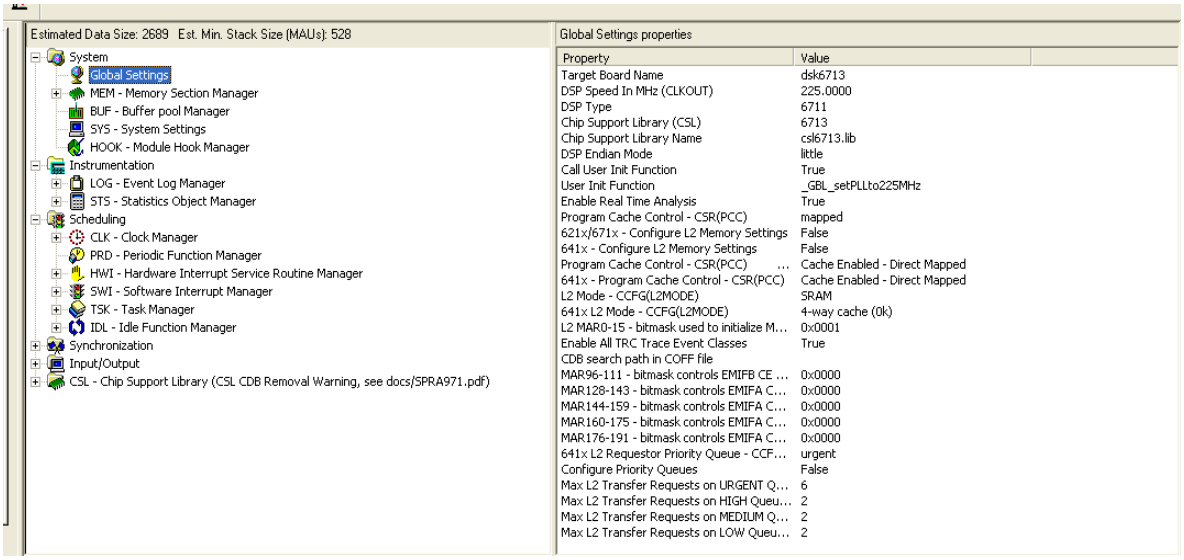
- Εισάγουμε στο πρόγραμμα μας το DSP /BIOS configuration. Το DSP /BIOS configuration περιέχει πληροφορίες για την κάρτα dsk671, που θα χρησιμοποιήσουμε στο project μας. Για να επιτευχθεί χρειαζόμαστε ένα πρότυπο dsk6713.cdb από πρόγραμμα.

  Ρυθμίσεις για εισαγωγή πρότυπου dsk6713.cdb:

  - Επιλέγουμε file –new- DSP /BIOS configuration και επιλέγουμε dsk6713(Σχήμα 3.8) και πατάμε OK.
  - Εμφανίζει ένα παράθυρο (σχήμα 3.9) Interrupt and System Configuration και κάνουμε έλεγχο στο GLOPAL SETTINGS αν είναι η κάρτα dsk6713 και χρησιμοποιεί την βιβλιοθήκη csl6713.lib. Με δεξί click στο properties κάνουμε τις ρυθμίσεις όπως φαίνεται στο ίδιο σχήμα.
  - Όταν τελειώσουμε αποθηκεύουμε ως led.cdb στο (C:\code composer studio 3.1 \ my project\ \led.c).

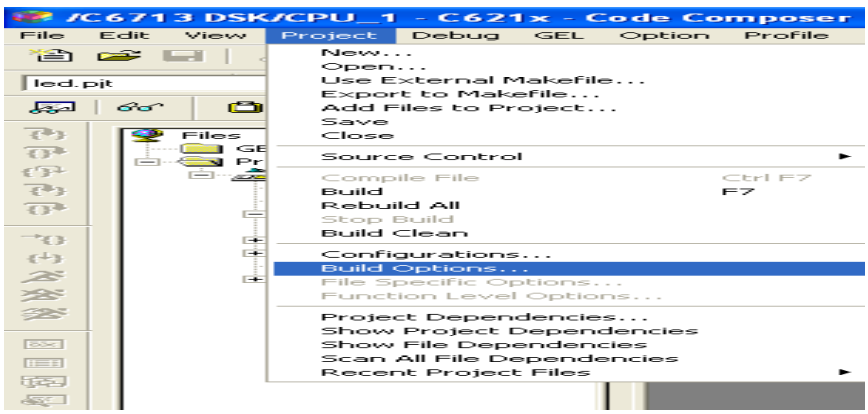


Σχήμα 3.8



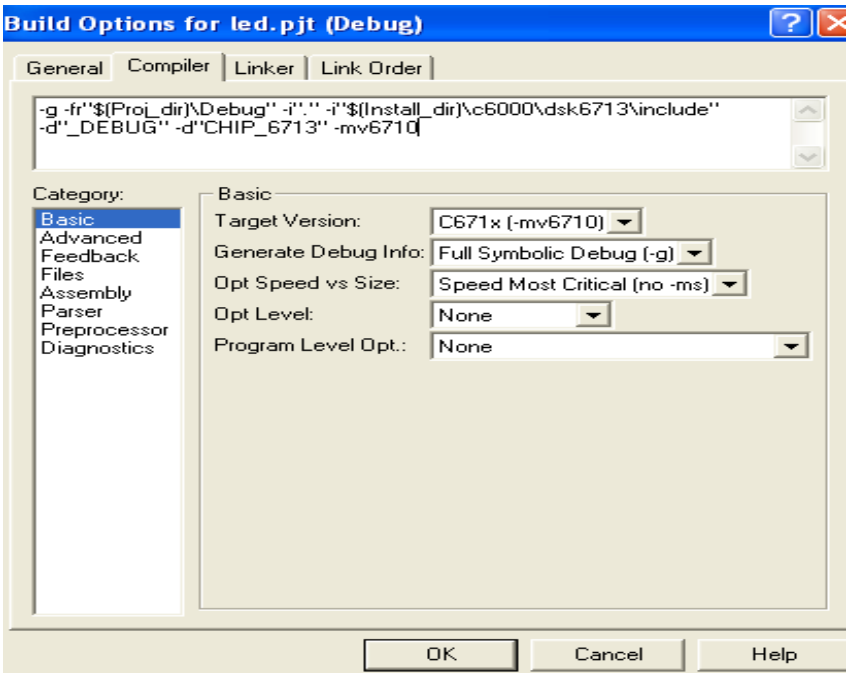
Σχήμα 3.9 (Interrupt and System Configuration).

- Ρυθμίζουμε τα Build Options στο πλαίσιο του project που οικοδομήσαμε όπου πρέπει να τροποποιηθούν ορισμένοι παράμετροι, όπως τα ακόλουθα σχήματα.
  - menu-project –build options.



σχήμα 3.10 (build-options)

- Στο target version η ενδεικτική απόδοση πρέπει να είναι c6713x.

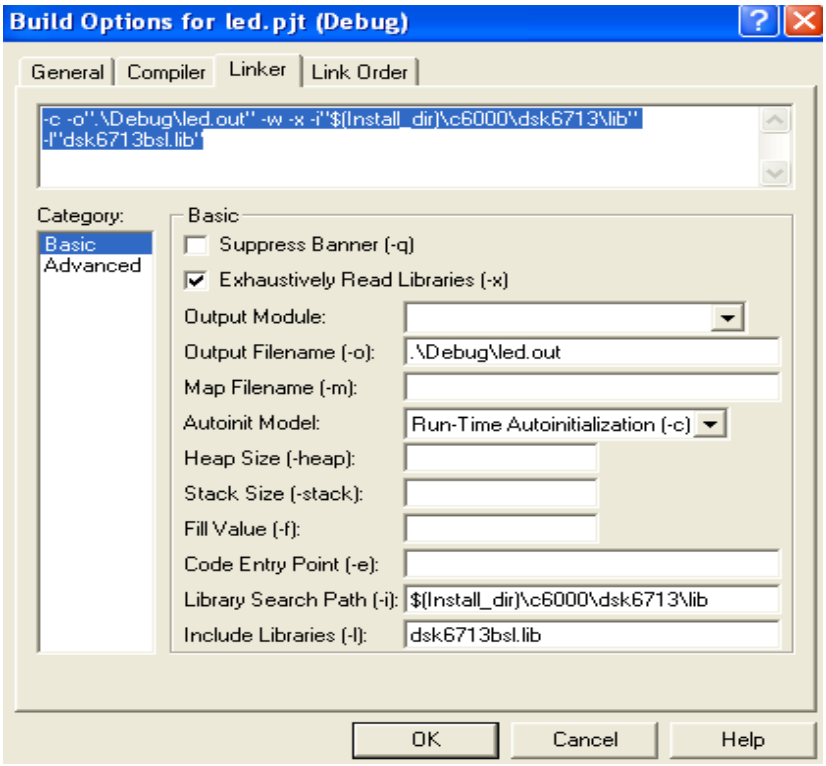


Σχήμα 3.11

- Το library search path πρέπει να είναι:

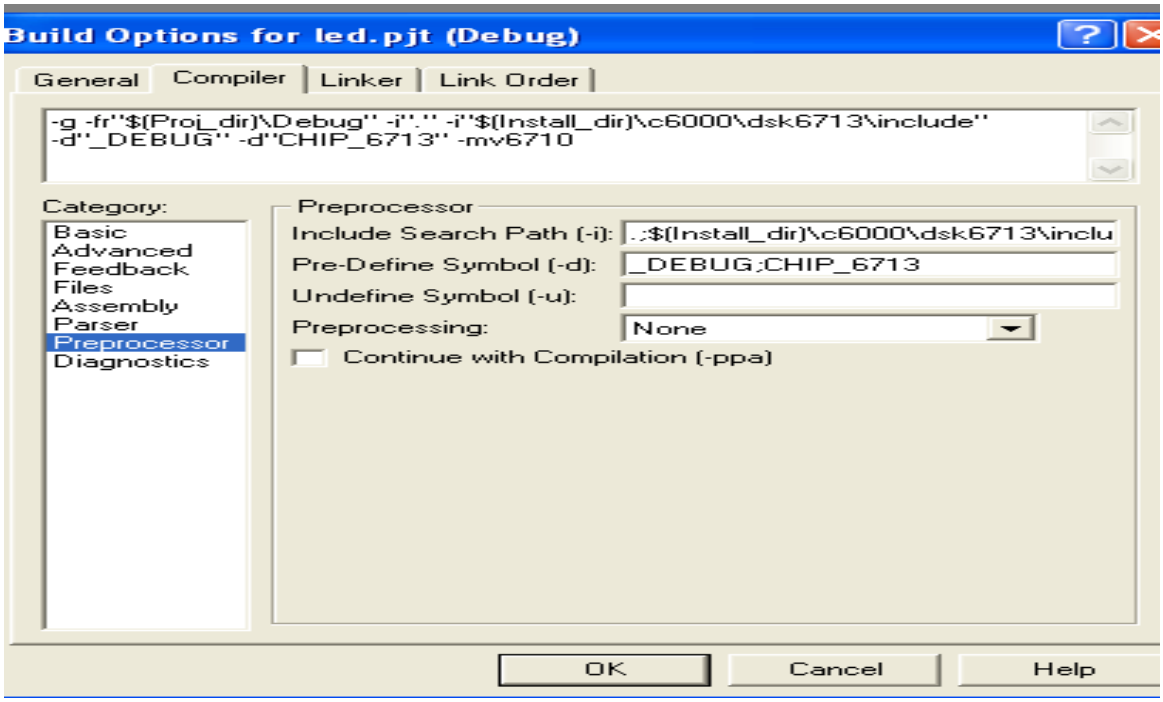
  .;$(install_dir)\c6000\dsk6713\include

  Το Inglude libraries πρέπει να είναι dsk6713bsl.lib



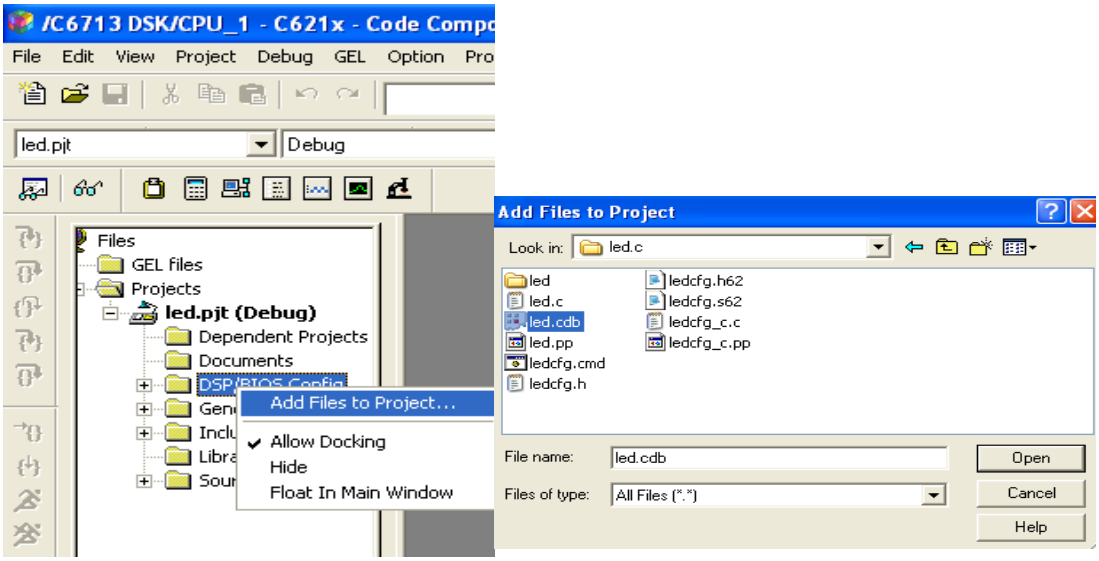
Σχήμα 3.12

- Στο pre-processor πρέπει να καθορίσουμε τα δύο σύμβολα. Στο (pre-define-symbol(-d) γράφουμε (_DEBUG;CHIP_6713),
  και στο (undefine symbol -κενό). Στο include search path ΄γράφουμε .;$(install_dir)\c6000\dsk6713\include.



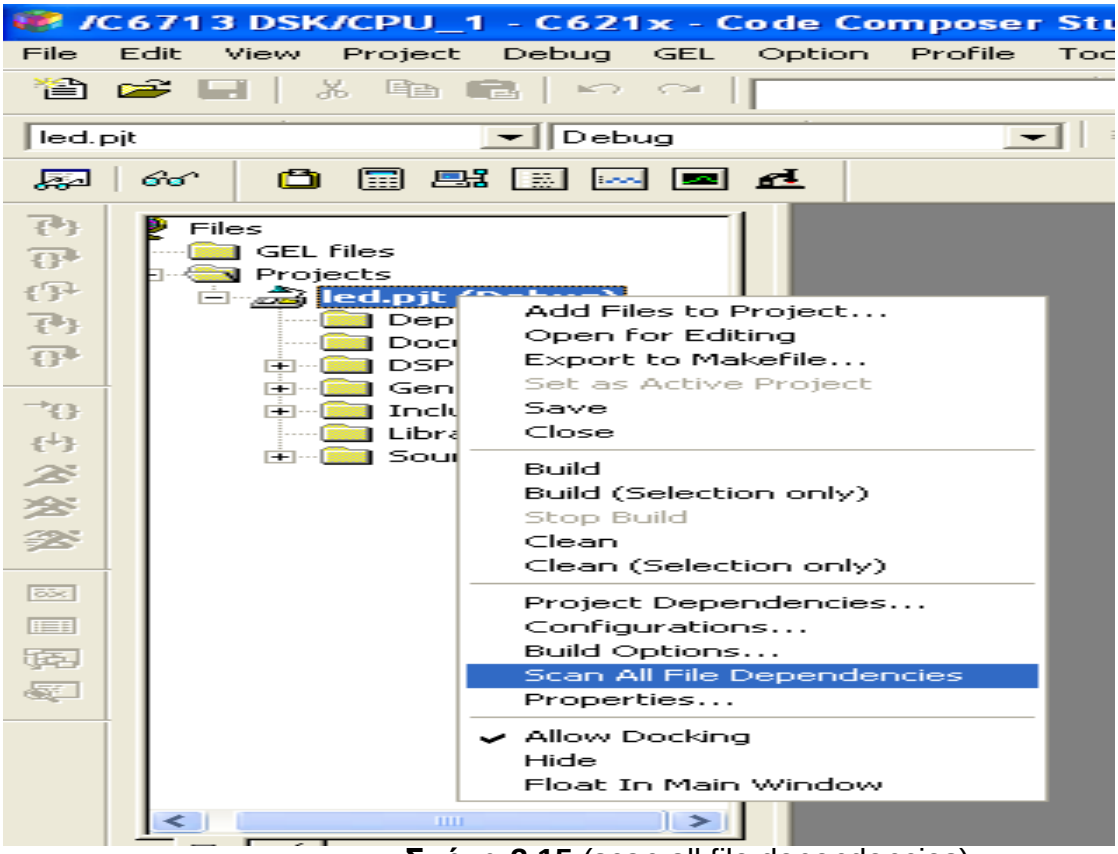
Σχήμα 3.13

- Εισαγωγή των αρχείων που δημιουργήσαμε στο project από το φάκελο που τα έχουμε αποθηκεύσει στο (C:\code composer studio 3.1 \ my project\ \led.c)
  - Menu- DSP/BIOS –δεξί click και add files –open.
    Menu -Source - δεξί click και add files-open.



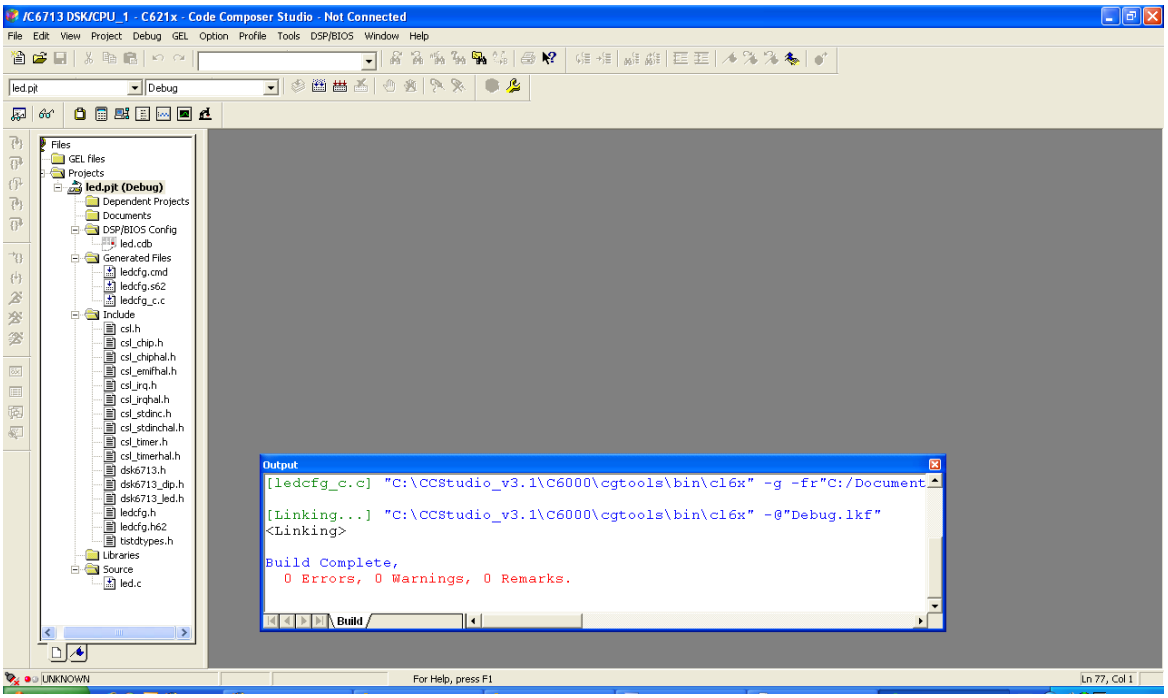
Σχήμα 3.14(add-files).

- Menu -Project scan all file dependencies



Σχήμα 3.15 (scan all file dependencies).

- Έλεγχος αν δουλευει το project .project-build.

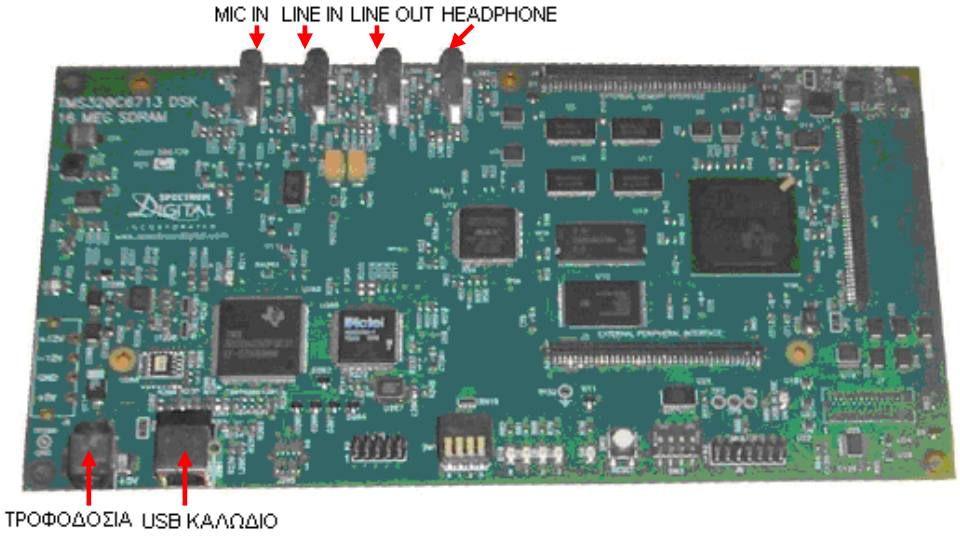


Σχήμα 3.16 (τελική μορφή project)

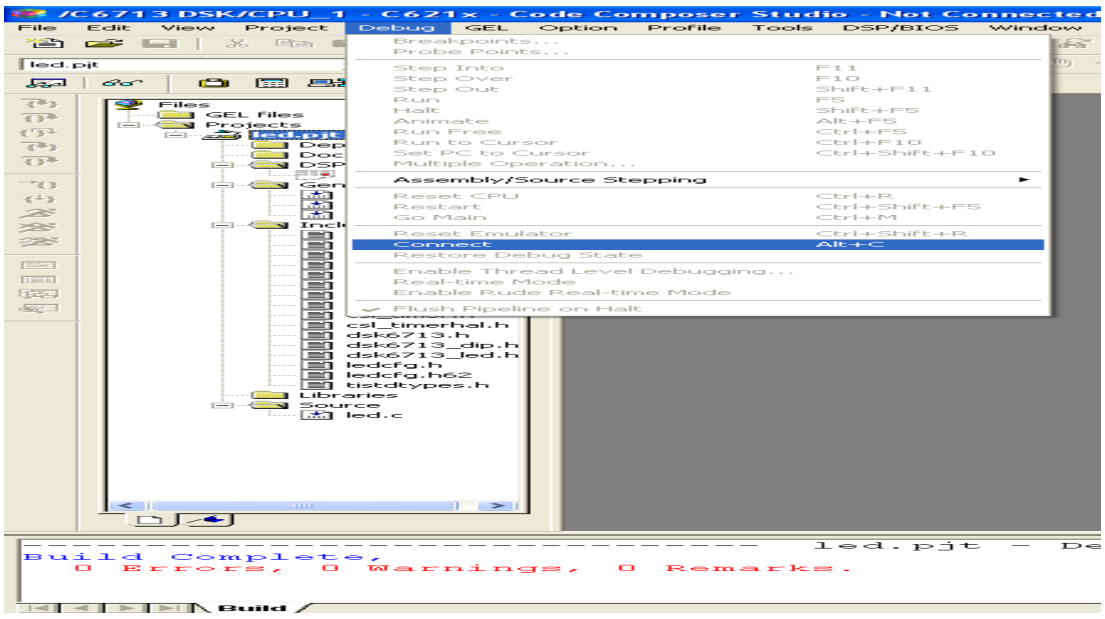
Βήμα 4

Συνδέουμε την κάρτα c6713 dsk με τον υπολογιστή και ρυθμίζουμε, σύμφωνα με τα βήματα:

- Συνδέουμε το DSK με το pc μέσω USB καλωδίου(σχήμα 3.17).
- Συνδέουμε την υποδοχή της τροφοδοσίας στο DSK(σχήμα 3.17).
- Συνδέουμε την line in με την γεννήτρια και line out με τον παλμογράφο (σχήμα 3.17).
- Με το άνοιγμα του υπολογιστή τρέχει το πρόγραμμα αυτόματου ελέγχου .
- Project –debug-connect (σχήμα 3.18).
- File –load program-debug out–run.



Σχήμα 3.17



Σχήμα 3.18

ΚΕΦΑΛΑΙΟ 4

Υπολογισμός Αλγόριθμων IIR φίλτρων

4.Αλγόριθμοι για συντελεστές IIR φίλτρων

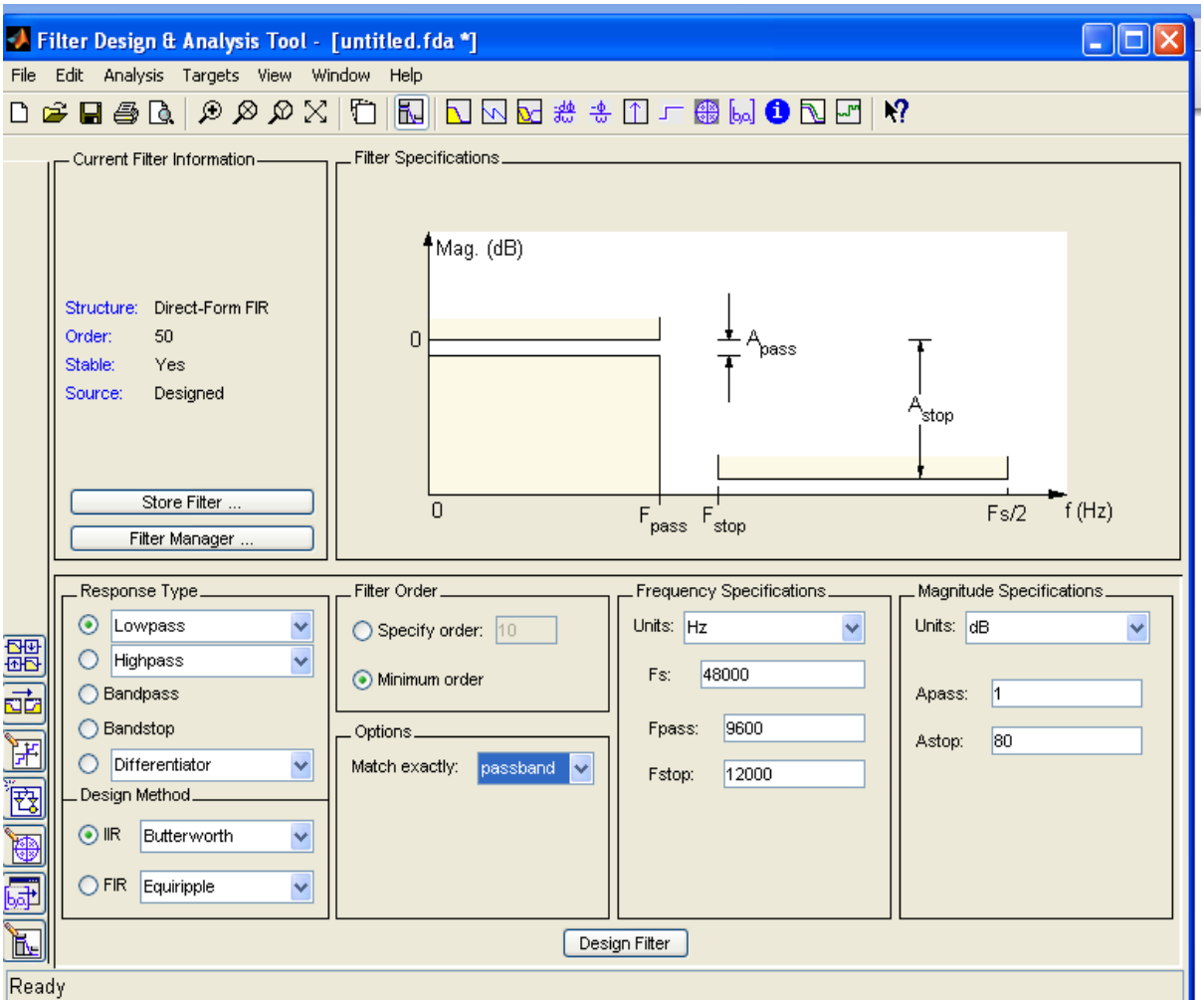
Οι αλγόριθμοι για συντελεστές IIR φίλτρων υπολογίζονται με τους ακόλουθους δύο τρόπους:

1. Με τη βοήθεια του matlab
2. Με τη βοήθεια του fdatool (filter design and analysis tool)

4.1 Λειτουργία fdatool

Το fdatool είναι ένα γραφικό περιβάλλον για σχεδίαση και ανάλυση των φίλτρων.

Η αρχική μορφή του fdatool φαίνεται στην εικόνα 4.1, η οποία θα μας βοηθήσει στην εξήγηση της λειτουργίας του.



Εικόνα 4.1

Βήματα σχεδίασης

1. Επιλέγουμε τον τύπο του φίλτρου στο πλαίσιο response type το τύπο του φίλτρου που θέλουμε να σχεδιάσουμε (low-pass,high-pass,band-pass και stop-band).
2. Επιλέγουμε τη μέθοδο σχεδίασης του φίλτρου στο πλαίσιο design method
   FIR φίλτρα με τον τύπο σχεδίασης (window,equiripple κ.τ.λ) ή
   IIR φίλτρα με τον τύπο σχεδίασης (Butterworth,chebyshev1 κ.λ.π).
3. Στο πλαίσιο filter orter καθορίζουμε την τάξη του φίλτρου στην επιλογή (spesity order)ή επιλέγουμε την επιλογή (minimum order)για να καθοριστεί η ελάχιστη τάξη του φίλτρου.
4. Στην επιλογή (option) προσδιορίζουμε τις επιλογές για τον τύπο του φίλτρου. Αυτή η επιλογή εξαρτάται από τον τύπο του φίλτρου δηλαδή αν είχαμε FIR φίλτρο με τη μέθοδο window θα επιλέγαμε τον τύπο του παραθύρου.
5. Στα πλαίσια (frequency specifications)και (magnitude specifications) καθορίζουμε τις προδιαγραφές των φίλτρων και το ρυθμό δειγματοληψίας.
6. Όταν τελειώσουμε με τα βήματα 1-5 πατάμε design filter και το φίλτρο θα σχεδιαστεί.
7. Η επιλογή (analysis –filter coefficients) μας εμφανίζει τους συντελεστές του φίλτρου.

4.1.1 Υπολογισμός αλγόριθμων για συντελεστές IIR φίλτρου με το fdatool

Εφαρμογή 1

Να υπολογιστούν οι αλγόριθμοι για συντελεστές IIR βαθυπερατού (low-pass)φίλτρου (Butterworth) με τις εξής προδιαγραφές :

Συχνότητα δειγματοληψίας: fs=48000hz;

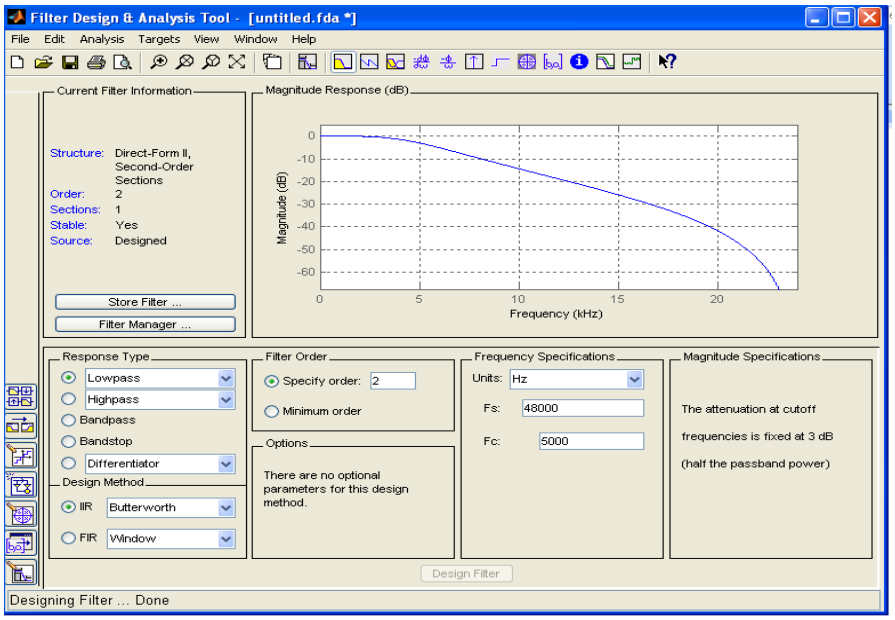
Συχνότητα αποκοπής: fc=5000hz;

Η τάξη του φίλτρου να είναι N=2;

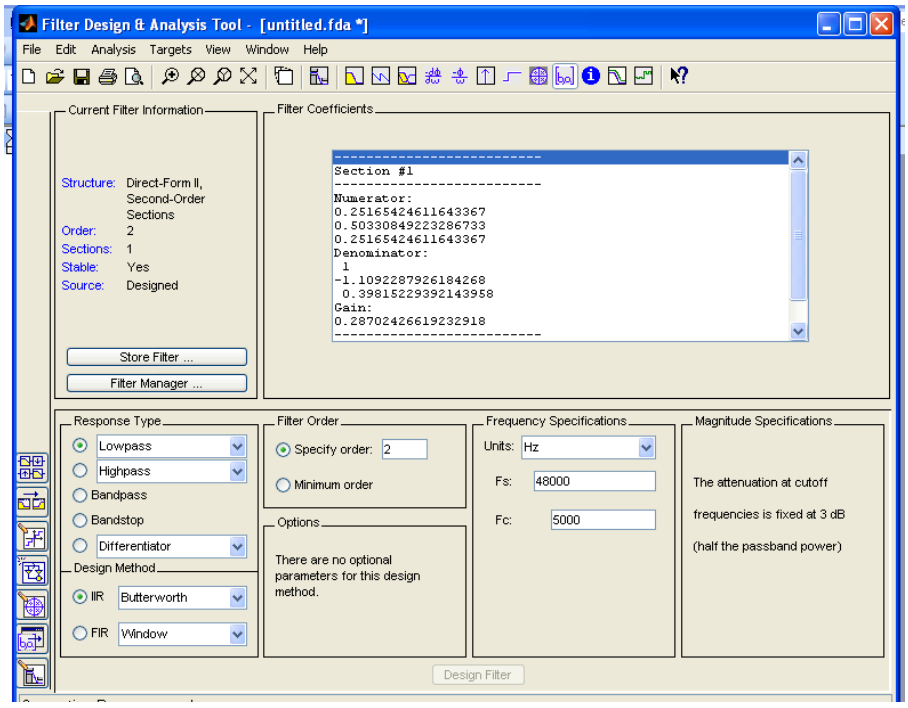
Βήματα υπολογισμού:

1. Ξεκινούμε το fdatool από το Matlab γράφοντας στο command window την εντολή >> fdatool.
2. Στο πλαίσιο response type επιλέγουμε low pass
3. Στο πλαίσιο design method επιλέγουμε IIR,και μετά butterworth.
4. Στο πλαίσιο filter order επιλέγουμε specify order και βάζουμε 2 που είναι η επιθυμητή τάξη.

5. Στα πλαίσια (frequency specifications)και (magnitude specifications)καθορίζουμε τις προδιαγραφές των φίλτρων και το ρυθμό δειγματοληψίας fs=48000hz,fc=5000hz;
6. Πατάμε design filter και το φίλτρο θα σχεδιαστεί.(εικόνα 4.2)
7. Η επιλογή (analysis –filter coefficients) μας εμφανίζει τους συντελεστές του φίλτρου (εικόνα 4.3)



Εικόνα 4.2



Εικόνα 4.3

4.2 Υπολογισμός αλγόριθμων για συντελεστές IIR φίλτρου με το Matlab.

Το ακόλουθο πρόγραμμα στο Matlab υπολογίζει τις σταθερές ενός IRR φίλτρου. Η τάξη του φίλτρου, η συχνότητα αποκοπής και η συχνότητα δειγματοληψίας, καθορίζονται με παραμέτρους μέσα στο πρόγραμμα.

Αφού υπολογιστούν οι σταθερές, γίνονται τα διαγράμματα απόκρισης-συχνότητας και διαφοράς φάσης-συχνότητας και αποθηκεύονται σε αρχείο .txt που ορίζεται από το χρήστη.

Εφαρμογή 2

Να υπολογιστούν οι αλγόριθμοι για συντελεστές IIR βαθυπερατού (low-pass)φίλτρου (Butterworth) με τις εξής προδιαγραφές :

Συχνότητα δειγματοληψίας: fs=48000hz;

Συχνότητα αποκοπής: fc=5000hz;

Η τάξη του φίλτρου να είναι N=2;

Βήματα υπολογισμού:

1. *Ορισμός παραμέτρων και υπολογισμός σταθερών του φίλτρου.*

ws=input('enter stop band edge freq: '); Ζητείται από το χρήστη να εισάγει την συχνότητα αποκοπής του φίλτρου σε HZ.

fs=input('enter the sampling freqency: '); Ζητείται από το χρήστη να εισάγει τη συχνότητα δειγματοληψίας του φίλτρου σε HZ.

wn=2*ws/fs; Η μεταβλητή αυτή έχει τιμή ίση με ω_c/π, όπου το ω_c δίνεται από τη σχέση $\omega_c=2*\pi*fp/fs$. Η τιμή αυτής της μεταβλητής χρησιμοποιείται ως παράμετρος στη συνάρτηση του Matlab, που υπολογίζει τις σταθερές iir.

2. *Υπολογισμός των σταθερών του φίλτρου.*

[b,a]=butter(2,wn); Σε αυτό το βήμα, γίνεται ο τελικός υπολογισμός των σταθερών του φίλτρου. Ο υπολογισμός γίνεται με τη συνάρτηση butter του Matlab, η οποία δέχεται τρία ορίσματα εισόδου, την τάξη του φίλτρου, την μεταβλητή wn που υπολογίστηκε στο προηγούμενο βήμα και τέλος ορίζεται ο τύπος του φίλτρου (στην περίπτωση μας έχουμε βαθυπερατό φίλτρο και δεν γράφουμε τίποτα. Όταν έχουμε υψηπερατό γράφουμε high, stop για φίλτρο αποκοπής ζώνης και bandpass για φίλτρο διέλευσης ζώνης). Η συγκεκριμένη συνάρτηση είναι για υπολογισμό φίλτρων butterworth. Άλλες συναρτήσεις ανάλογα με τον τύπο του φίλτρου που σχεδιάζεται είναι besself,cheby1,cheby2. Για περισσότερες πληροφορίες σχετικά με αυτές τις συναρτήσεις μπορούν να δοθούν με την εντολή **>>help (όνομα συνάρτησης)** από τη γραμμή εντολών του Matlab.

3. *Σχεδίαση γραφικών παραστάσεων απόκρισης – συχνοτήτων και φάσης – συχνότητας.*

[h,omega]=freqz(b,a); %Στο σημείο αυτό χρησιμοποιείται η συνάρτηση freqz με ορίσματα τον πίνακα με τις σταθερές του φίλτρου b. Στα fir φίλτρα υπάρχουν μόνο οι σταθερές b, οι σταθερές του αριθμητή (numerators), στη σχέση που συνδέει την απόκριση του φίλτρου με τις σταθερές a και b. Το omega, το οποίο είναι το διάνυσμα συχνοτήτων σε radians/sample, παίρνει τιμές στο διάνυσμα [0, π]. Αυτό σημαίνει ότι το διάστημα από [0, π] των συχνοτήτων, χωρίζεται σε ίσα μέρη των οποίων οι τιμές αποθηκεύονται στο διάνυσμα omega. Επειδή δεν ορίζεται σε πόσα μέρη θα χωριστεί το διάστημα αυτό, ισχύει η προκαθορισμένη τιμή που είναι 512, δηλαδή το διάστημα αυτό θα χωριστεί σε 512 ίσα μέρη. Τα διανύσματα h και omega θα έχουν 512 τιμές. Για κάθε τιμή του omega υπολογίζεται μία τιμή απόκρισης, με βάση τον ακόλουθο μαθηματικό τύπο, η οποία αποθηκεύεται στο διάνυσμα h. Τα ζεύγη των τιμών h και omega είναι που θα δώσουν την γραφική παράσταση της απόκρισης.

$$H(e^{jw}) = \frac{B(e^{jw})}{A(e^{jw})} = \frac{b(1) + b(2)e^{-jw} + \dots + b(m+1)e^{-jmw}}{a(1) + a(2)e^{-jw} + \dots + a(n+1)e^{-jnw}}$$

gain=20*log10(abs(h));% Απο το διάνυσμα της απόκρισης του φίλτρου h υπολογίζεται το διάνυσμα της απολαβής gain.

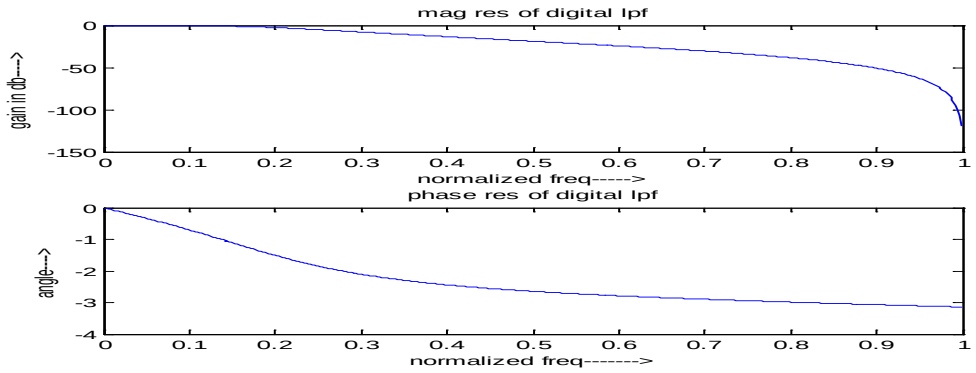
an=angle(h);Υπολογίζεται το διάνυσμα της διαφοράς φάσης.

Στα επόμενα βήματα σχεδιάζονται οι γραφικές παραστάσεις απολαβής – συχνότητας και διαφοράς φάσης – συχνότητας. Το διάνυσμα omega συχνοτήτων κανονικοποιείται omega/pi έτσι ώστε να παίρνει τιμές στο διάστημα [0, 1]

```
subplot(2,1,1);
plot(omega/pi,gain);
title('mag res of digital hpf');
xlabel('normalized freq----->');
ylabel('gain in db----->')
subplot(2,1,2);
plot(omega/pi,an);
title('phase res of digital hpf');
xlabel('normalized freq------->');
```

ylabel('angle---->');

Το αποτέλεσμα της σχεδίασης των γραφικών παραστάσεων φαίνεται στο ακόλουθο σχήμα:



Σχήμα 4.1

4. *Κβάντιση σταθερών.*

Εδώ γίνεται η κβάντιση των σταθερών a και b. Μετά από αυτή τη διαδικασία οι σταθερές είναι ακέραιες και παίρνουν τιμές στο διάστημα [0, 2^15= 32768] .

```
a=a.*2^15;
b=b.*2^15;
a=round(a);
b=round(b);
a(2)=round(a(2)./2);
b(2)=round(b(2)./2);
```

5. *Αποθήκευση σταθερών σε αρχείο κειμένου.*

```
[filename,pathname,filterindex]=uiputfile('*.txt','Save the fir cofficients');
fid=fopen([pathname,filename],'w');
fprintf(fid,'/* FIR coefficients in float format */\r\n');
for i=1:1:length(b)
   fprintf(fid,'%d,',b(i));
end
for i=1:1:length(a)-1
   fprintf(fid,'%d,',a(i));
end
fprintf(fid,'%d',a(length(a)));
fclose(fid);
/* IIR coefficients in float format */
2367,2367,2367,32768,-18174,13047
```

4.3 Αλγόριθμοι ψηφιοποίησης IIR φίλτρου

Για την υλοποίηση της ψηφιοποίησης των αλγόριθμων των IIR φίλτρων χρησιμοποιήσαμε ένα έτοιμο πρόγραμμα γραμμένο σε γλώσσα C της (Texas Instrument) και η αλλαγή που κάναμε ήταν οι αλγόριθμοι των συντελεστών των φίλτρων που βγάλαμε από το πρόγραμμα στο Matlab(4.2.1)και fdatool(4.1.2)ανάλογα με τον τύπο του φίλτρου οι συντελεστές άλλαζαν .

Το πρόγραμμα τις (Texas Instrument) φαίνεται πιο κάτω:

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N =2  //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[]={
//2367,2367,2367,32768,-18174,13047/*LPF 5000 */
} ;
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
```

Κυρίως πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληψίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.

```
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοποίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Έναρξη ατέρμονος βρόχου.
{ /* Διάβασε ένα δείγμα από το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διάβασε ένα δείγμα από το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου από το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητή l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσιμο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
```

```
/*Αλγόριθμος φιλτραρίσματος. Υλοποιεί φίλτρα δεύτερης τάξης*/
signed int IIR_FILTER(const signed int * h, signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Πρέπει να δηλωθεί ως τύπου static, ώστε να διατηρεί τις τιμές που είχε από προηγούμενες κλήσεις της συνάρτησης */static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Πρέπει να δηλωθεί ως τύπου static, ώστε να διατηρεί τις τιμές που είχε από προηγούμενες κλήσεις της συνάρτησης . Στον πίνακα y αποθηκεύονται φιλτραρισμένες τιμές από προηγούμενα βήματα. Οι τιμές αυτές χρησιμοποιούνται στη διαδικασία φιλτραρίσματος σε επόμενα δείγματα εισόδου*/
int temp=0; /*Αρχικοποίηση του temp*/
temp = (short int)x1; /* Αντιγραφή του δείγματος εισόδου στη μεταβλητή temp */
x[0] = (signed int) temp; /* Αντιγραφή της μεταβλητής temp στην πρώτη θέση του πίνακα x. Το χ(0) έχει τώρα την τιμή του δείγματος εισόδου x[stages][0]. Στη συνέχεια ακολουθούν τα βήματα για τον υπολογισμό του φιλτραρισμένου δείγματος εξόδου, το οποίο αποθηκεύεται στην μεταβλητή temp */
temp = ( (int)h[0] * x[0]) ; /* B0 * x(n) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[2] * x[2]); /* B2 * x(n-2) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */
/* Στο σημείο αυτό του κώδικα γίνεται η αντίστροφη διαδικασία από αυτή της κβάντισης, που είδαμε στο πρόγραμμα του Matlab */
temp >>= 15; /* Ολίσθηση κατά 15 θέσεις δεξιά, αντιστοιχεί με διαίρεση με το 2^15*/
if ( temp > 32767 )
{
temp = 32767;
}
else if ( temp < -32767)
{
temp = -32767;
}
```

```
y[0] = temp ;/* ο φιλτραρισμένο δείγμα αποθηκεύεται στην πρώτη θέση του πίνακα y*/
/* Ολίσθηση όλων των στοιχείων στους πίνακες x και y κατά 1 θέση δεξιά*/
y[2] = y[1]; /* y(n-2) = y(n-1) */
y[1] = y[0]; /* y(n-1) = y(n) */
x[2] = x[1]; /* x(n-2) = x(n-1) */
x[1] = x[0]; /* x(n-1) = x(n) */
/* temp is used as input next time through */
return (temp<<1);
}
```

4.4 Ολοκλήρωση προγράμματος C και κατέβασμα στο DSP

Όταν ολοκληρώσουμε τα πιο πάνω βήματα δηλαδή βρούμε τους αλγόριθμους των συντελεστών του φίλτρου και τους εισάγουμε στο πρόγραμμα της (Texas Instrument) το επόμενο βήμα είναι να εισάγουμε το πρόγραμμα τις C σε ένα project που δημιουργήσαμε στο code composer studio (κεφάλαιο 3.3.1).

Για την ολοκλήρωση του προγράμματος και κατέβασμα στο dsp πρέπει όταν εκτελέσουμε το project κατά την διαδικασία project - build πρέπει να μας δείξει το πρόγραμμα (C.C.S)Build –complete ότι το πρόγραμμα της C εκτελείται .

Το επόμενο βήμα είναι να κατεβάσουμε το πρόγραμμα στο dsp

- Project –debug-connect (σχήμα 3.18).
- File –load program-debug out–run.

ΚΕΦΑΛΑΙΟ 5

Υπολογισμός Αλγόριθμων FIR φίλτρων

5. Αλγόριθμοι για συντελεστές FIR φίλτρων

Οι αλγόριθμοι για συντελεστές FIR φίλτρων υπολογίζονται με δύο τρόπους .

1. Με τη βοήθεια του Matlab
2. Με τη βοήθεια του fdatool(filter design and analysis tool)

5.1 Υπολογισμός αλγόριθμων για συντελεστές FIR φίλτρου με το fdatool

Εφαρμογή 3

Να υπολογιστούν οι αλγόριθμοι για συντελεστές FIR βαθυπερατού (low-pass)φίλτρου με τη μέθοδο παραθύρων (window)με το παράθυρο (Blackman)με τις εξής προδιαγραφές :

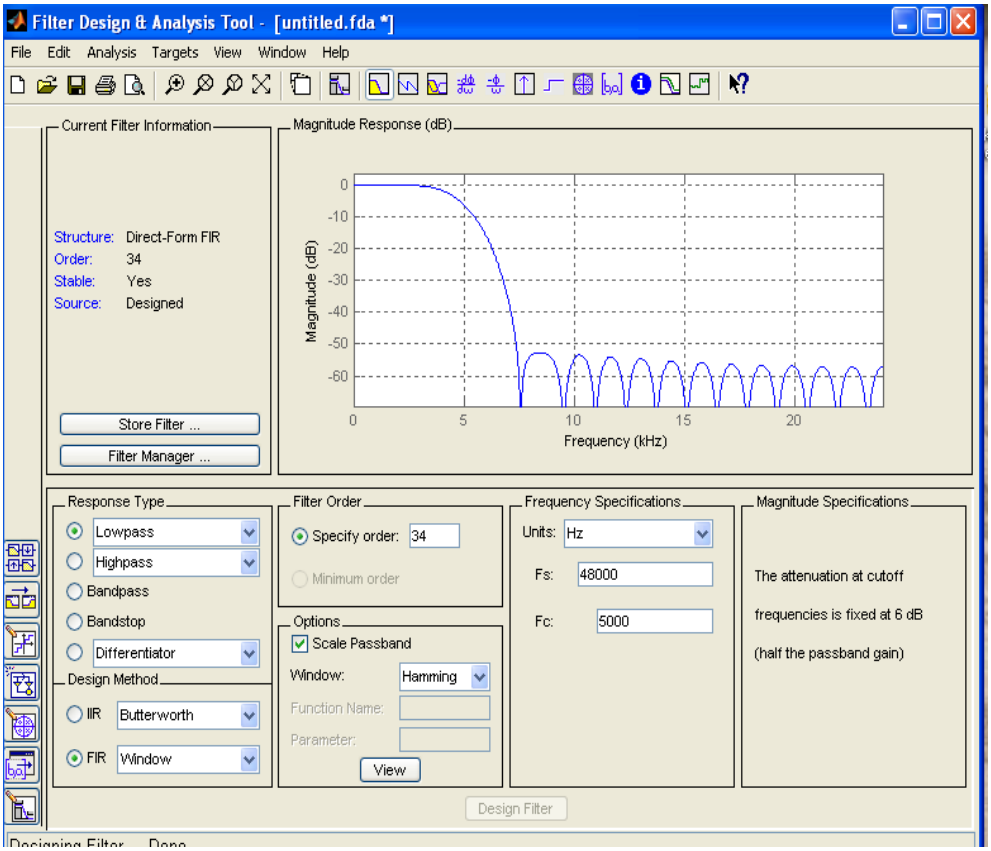
Συχνότητα δειγματοληψίας: fs=48000hz;

Συχνότητα αποκοπής: fc=5000hz;

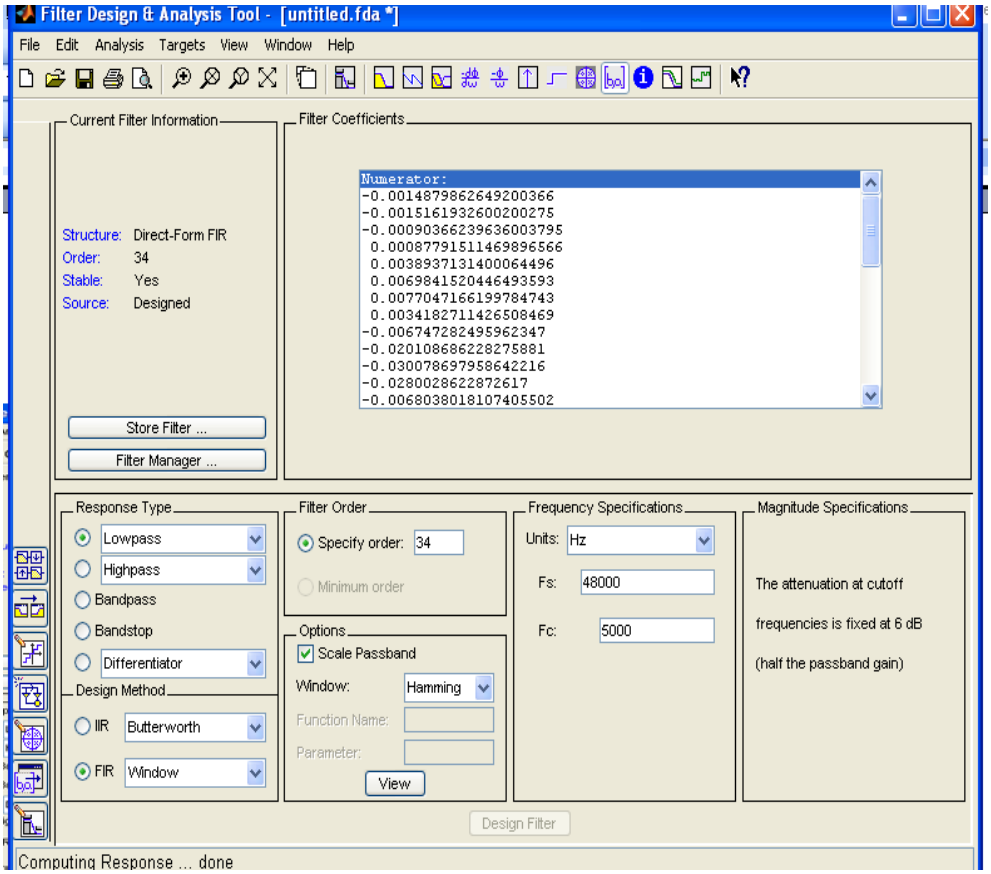
Η τάξη του φίλτρου να είναι N=34;

Βήματα υπολογισμού:

1. Ξεκινούμε το fdatool από το Matlab γράφοντας στο command window την εντολή >> fdatool.
2. Στο πλαίσιο response type επιλέγουμε low pass
3. Στο πλαίσιο design method επιλέγουμε FIR τον τύπο μετά τη μέθοδο σχεδίασης (window)και στο πλαίσιο order το παράθυρο (hamming) .
4. Στο πλαίσιο filter order επιλέγουμε specify order και βάζουμε 34 που είναι η επιθυμητή τάξη.
5. Στα πλαίσια (frequency specifications)και (magnitude specifications)καθορίζουμε τις προδιαγραφές των φίλτρων και το ρυθμό δειγματοληψίας fs=48000hz,fc=5000hz;
6. Πατάμε design filter και το φίλτρο θα σχεδιαστεί.(εικόνα 5.1)
7. Η επιλογή (analysis –filter coefficients) μας εμφανίζει τους συντελεστές του φίλτρου (εικόνα 5.2)



Εικόνα 5.1



Εικόνα 5.2

5.2 Υπολογισμός αλγόριθμων για συντελεστές FIR φίλτρου με το Matlab.

Το ακόλουθο πρόγραμμα στο Matlab υπολογίζει τις σταθερές ενός FIR φίλτρου. Η τάξη του φίλτρου, η συχνότητα αποκοπής και η συχνότητα δειγματοληψίας, καθορίζονται με παραμέτρους μέσα στο πρόγραμμα.

Αφού υπολογιστούν οι σταθερές, γίνονται τα διαγράμματα απόκρισης-συχνότητας και διαφοράς φάσης-συχνότητας και αποθηκεύονται σε αρχείο .txt που ορίζεται από το χρήστη.

Υπολογισμός αλγόριθμων για συντελεστές FIR φίλτρου με το Matlab

Εφαρμογή 4

Να υπολογιστούν οι αλγόριθμοι για συντελεστές FIR βαθυπερατού (low-pass)φίλτρου με τη μέθοδο παραθύρων (window)με το παράθυρο (Blackman)με τις εξής προδιαγραφές :

Συχνότητα δειγματοληψίας: fs=48000hz;

Συχνότητα αποκοπής: fc=5000hz;

Η τάξη του φίλτρου να είναι N=34;

Βήματα υπολογισμού:

1. *Ορισμός των παραμέτρων του προγράμματος*

```
n=34;
%Στη μεταβλητή n ορίζεται η τάξη του φίλτρου.
fc=5000;
%Στη μεταβλητή fp ορίζεται η συχνότητα αποκοπής του φίλτρου σε HZ. Η συχνότητα αυτή δεν πρέπει να είναι μεγαλύτερη από το μισό της συχνότητας δειγματοληψίας fs, δηλαδή μεγαλύτερη από fs/2.
 fs=48000;
%Στη μεταβλητή αυτή ορίζεται η συχνότητα δειγματοληψίας του DSP. Στη περίπτωση που η συχνότητα δειγματοληψίας είναι 48000HZ, τότε τα υλοποιήσιμα φίλτρα, μπορούν να έχουν συχνότητα αποκοπής στο διάστημα από (0HZ, 22000HZ).
```

Στο σημείο αυτό έχει ολοκληρωθεί ο ορισμός των παραμέτρων του προγράμματος και ακολουθούν τα βήματα για τον υπολογισμό των σταθερών του φίλτρου.

```
fn=2*fc/fs;
```

Η μεταβλητή αυτή έχει τιμή ίση με ωc/π, όπου το ωc δίνεται από τη σχέση ωc=2*π*fc/fs. Η τιμή αυτής της μεταβλητής χρησιμοποιείται ως παράμετρος στη συνάρτηση του Matlab, που υπολογίζει τις σταθερές fir.

2. *Υπολογισμός των σταθερών του φίλτρου*

b=fir1(n,fn,window);

%Σε αυτό το βήμα, γίνεται ο τελικός υπολογισμός των σταθερών του φίλτρου. Ο υπολογισμός γίνεται με τη συνάρτηση fir1 η οποία παίρνει ορίσματα τις τιμές των μεταβλητών n, fn και window και επιστρέφει στο διάνυσμα b τις σταθερές του φίλτρου. Με τις συγκεκριμένες παραμέτρους εισόδου το φίλτρο που προκύπτει είναι βεθυπερατό. Αν θέλαμε να υπολογίσουμε σταθερές για υψηπερατό φίλτρο τότε η εντολή που θα γράφαμε σε αυτό το βήμα θα ήταν **b=fir1(n,fn,'high',window);** όπου η παράμετρος high προσδιορίζει τον τύπο του φίλτρου. Αντίστοιχα θα είχαμε stop για φίλτρο αποκοπής ζώνης και bandpass για φίλτρο διέλευσης ζώνης.

3. *Σχεδίαση γραφικών παραστάσεων απόκρισης – συχνοτήτων και φάσης – συχνότητας.*

Σε αυτό το τμήμα του προγράμματος γίνεται η σχεδίαση των γραφικών παραστάσεων της απόκρισης και της διαφοράς φάσης του φίλτρου. Σε αυτές τις γραφικές παραστάσεις ο άξονας των συχνοτήτων είναι κανονικοποιημένος στο διάστημα [0, 1]. Η κανονικοποίηση γίνεται ως προς την τιμή fs/2, αυτό σημαίνει ότι η τιμή 1 στο κανονικοποιημένο διάστημα τιμών αντιστοιχεί σε τιμή συχνότητας ίση με fs/2.

[H W]=freqz(b,1,128);

%Στο σημείο αυτό χρησιμοποιείται η συνάρτηση freqz με ορίσματα τον πίνακα με τις σταθερές του φίλτρου b. Στα fir φίλτρα υπάρχουν μόνο οι σταθερές b, οι σταθερές του αριθμητή (numerators), στη σχέση που συνδέει την απόκριση του φίλτρου με τις σταθερές a και b. Σταθερές a του παρονομαστή (denominators) δεν υπάρχουν και για το λόγο αυτό στη θέση τους μπαίνει 1. Το 128 είναι το πλήθος των τιμών απόκρισης H που θα υπολογιστούν μέσα στο διάνυσμα συχνοτήτων W, το οποίο είναι σε radians/sample και παίρνει τιμές στο διάνυσμα [0, π]. Αυτό σημαίνει ότι το διάστημα από [0, π] των συχνοτήτων, χωρίζεται σε 128 ίσα μέρη των οποίων οι τιμές αποθηκεύονται στο διάνυσμα W. Για κάθε τιμή του W υπολογίζεται μία τιμή απόκρισης, με βάση τον ακόλουθο μαθηματικό τύπο, η οποία αποθηκεύεται στο διάνυσμα H. Τα ζεύγη των τιμών H και W είναι που θα δώσουν την γραφική παράσταση της απόκρισης.

$$H(e^{jw}) = \frac{B(e^{jw})}{A(e^{jw})} = \frac{b(1) + b(2)e^{-jw} + \dots + b(m+1)e^{-jmw}}{a(1) + a(2)e^{-jw} + \dots + a(n+1)e^{-jnw}}$$

subplot(2,1,1);

% Η εντολή αυτή σημαίνει ότι στο ίδιο figure που θα ανοίξει το Matlab, θα υπάρχουν 2 γραφικές παραστάσεις, η μία κάτω από την άλλη. Η μία γραφική παράσταση είναι συχνοτική απόκριση του φίλτρου και η άλλη είναι το διάγραμμα της διαφοράς φάσης.

plot(W/pi,abs(H));

% Σε αυτό το βήμα σχεδιάζεται η πρώτη γραφική παράσταση.

title('magnitude response of lpf');

% Δίνεται τίτλος στη γραφική παράσταση. Δίνονται τίτλοι στους άξονες X και Y

ylabel('gain in db-------->');

xlabel('normalized frequency------>');

subplot(2,1,2);%Θα γίνει σχεδίαση δεύτερης γραφικής παράστασης στο ίδιο Figure

plot(W/pi,angle(H));%Σχεδιάζεται η δεύτερη γραφική παράσταση κάτω από την πρώτη

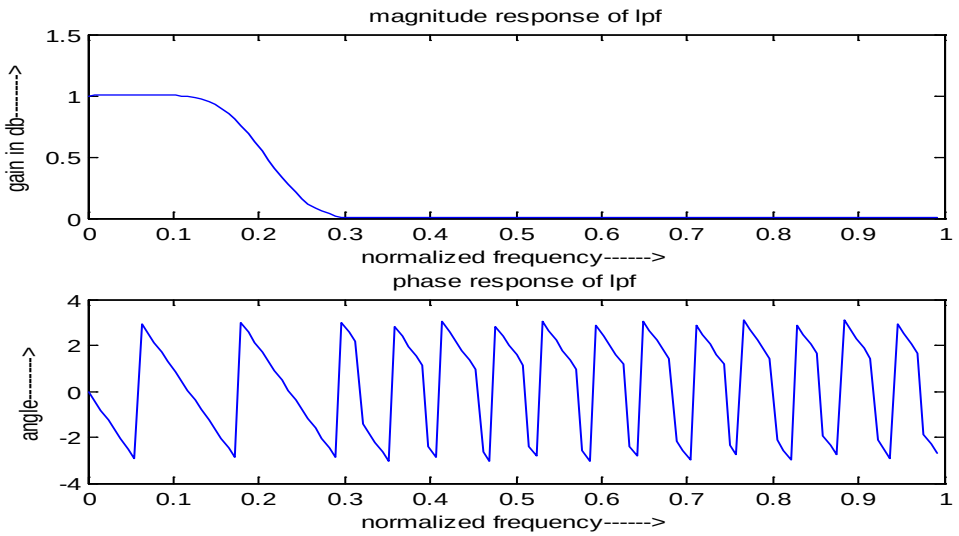
Δίνονται τίτλοι στη γραφική παράσταση και στους άξονες.

title('phase response of lpf');

ylabel('angle-------->');

xlabel('normalized frequency------>');

Το αποτέλεσμα του κώδικα αυτού φαίνεται στην ακόλουθη εικόνα:



Εικόνα 5.3

4. *Αποθήκευση σταθερών σε αρχείο κειμένου.*

[filename,pathname,filterindex]=uiputfile('*.txt','Save the fir cofficients');

% Με τη συνάρτηση uiputfile ανοίγει ένα γραφικό περιβάλλον από το Matlab που ζητάει από το χρήστη να ορίσει το όνομα, τη διαδρομή και τον τύπο του αρχείου. Αποθηκεύει τις επιλογές του στα διανύσματα filename,pathname και filterindex.

fid=fopen([pathname,filename],'w');

% Η fopen δημιουργεί το καινούριο αρχείο με δυνατότητα εγγραφής δεδομένων σε αυτό, αυτό ορίζεται από την παράμετρο ‘w’ (write). Επιστρέφει επίσης έναν δείκτη σε αυτό το αρχείο, τον fid, μέσω του οποίου μπορούν να γίνονται αναφορές στο αρχείο για εγγραφή δεδομένων από τη συνάρτηση fprintf.

```
fprintf(fid,'/* FIR coefficients in float format */\r\n');
counter=1;
fprintf(fid,'{\r\n');
for i=1:1:length(b)
if (i~=length(b))
fprintf(fid,'%d,',b(i));
counter=counter+1;
if (counter==5)
fprintf(fid,'\r\n');
counter=1;
end
else
fprintf(fid,'%d',b(i));
end
end
fprintf(fid,'\r\n}');
fclose(fid);%
/* FIR coefficients in float format */
{
-1.487986e-003,-1.516193e-003,-9.036624e-004,8.779151e-004,
3.893713e-003,6.984152e-003,7.704717e-003,3.418271e-003,
-6.747282e-003,-2.010869e-002,-3.007870e-002,-2.800286e-002,
-6.803802e-003,3.507958e-002,9.144915e-002,1.492446e-001,
1.926289e-001,2.087363e-001,1.926289e-001,1.492446e-001,
9.144915e-002,3.507958e-002,-6.803802e-003,-2.800286e-002,
-3.007870e-002,-2.010869e-002,-6.747282e-003,3.418271e-003,
7.704717e-003,6.984152e-003,3.893713e-003,8.779151e-004,
-9.036624e-004,-1.516193e-003,-1.487986e-003
}
```

5.3 Αλγόριθμοι ψηφιοποίησης FIR φίλτρου

Για την υλοποίηση της ψηφιοποίησης των αλγόριθμων των IIR φίλτρων χρησιμοποιήσαμε ένα έτοιμο πρόγραμμα γραμμένο σε γλώσσα C της (Texas Instrument) και η αλλαγή που κάναμε ήταν οι αλγόριθμοι των συντελεστών των φίλτρων που βγάλαμε από το πρόγραμμα στο Matlab(5.2.1) και fdatool(5.1.1) ανάλογα με τον τύπο του φίλτρου οι συντελεστές άλλαζαν.

Το πρόγραμμα τις (Texas Instrument) φαίνεται πιο κάτω:

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 34  //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[] ={
-1.487986e-003,-1.516193e-003,-9.036624e-004,8.779151e-004,
3.893713e-003,6.984152e-003,7.704717e-003,3.418271e-003,
-6.747282e-003,-2.010869e-002,-3.007870e-002,-2.800286e-002,
-6.803802e-003,3.507958e-002,9.144915e-002,1.492446e-001,
1.926289e-001,2.087363e-001,1.926289e-001,1.492446e-001,
9.144915e-002,3.507958e-002,-6.803802e-003,-2.800286e-002,
-3.007870e-002,-2.010869e-002,-6.747282e-003,3.418271e-003,
7.704717e-003,6.984152e-003,3.893713e-003,8.779151e-004,
-9.036624e-004,-1.516193e-003,-1.487986e-003
};
static short in_buffer[N];  /*Πίνακας που έχει καθολική εμβέλεια μέσα στο πρόγραμμα
και                          έχει οριστεί ως static έτσι ώστε να διατηρεί τιμές που
                             αποθηκεύονται σε αυτόν σε προηγούμενες επαναλήψεις του
                             προγράμματος*/
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα
στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την
αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
```

```
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate
control */\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
```

Κυρίως πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληψίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.

```
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοποίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα
δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Έναρξη ατέρμονος βρόχου.
{ /* Διάβασε ένα δείγμα από το αριστερό κανάλι. Αποθήκευσε την τιμή του στη
μεταβλητή l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διάβασε ένα δείγμα από το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή
r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου από το αριστερό κανάλι και αποθήκευση της
φιλτραρισμένης τιμής του στη μεταβλητή l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
```

```
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσιμο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος.*/
Ο αλγόριθμος αυτός υλοποιεί το άθροισμα
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* Πάντα κάθε νέο δείγμα εισόδου, πρίν τη διαδικασία φιλτραρισματος
                     τοποθετείται στην πρώτη θέση του πίακα in_buffer (in_buffer[0]) */
for(i=(N-1);i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* Μετακίνηση όλων των δειγμάτων κατα μία θέση δεξιά */
for(i=0;i<(N);i++)
output = output + h[i] * in_buffer[i];/*Υλοποιηση του παραπάνω αθροισματος */
return(output<<1);
{
```

5.4. Ολοκλήρωση προγράμματος C και κατέβασμα στο DSP

Όταν ολοκληρώσουμε τα πιο πάνω βήματα, δηλαδή βρούμε τους αλγόριθμους των συντελεστών του φίλτρου και τους εισάγουμε στο πρόγραμμα της (Texas Instrument) το επόμενο βήμα είναι να εισάγουμε το πρόγραμμα γλώσσας C σε ένα project που δημιουργήσαμε στο code composer studio (κεφάλαιο 3.4).

Για την ολοκλήρωση του προγράμματος και κατέβασμα στο dsp πρέπει όταν εκτελέσουμε το project κατά την διαδικασία project - build πρέπει να μας δείξει το πρόγραμμα (C.C.S)Build –complete ότι το πρόγραμμα της C εκτελείται.

Το επόμενο βήμα είναι να κατεβάσουμε το πρόγραμμα στο dsp.

- Project –debug-connect (σχήμα 3.18).
- File –load program-debug out–run.

ΚΕΦΑΛΑΙΟ 6

Υπολογισμός συνάρτησης μεταφοράς FIR και IIR φίλτρου

6.1 Υπολογισμός συνάρτησης μεταφοράς FIR φίλτρου

Παράδειγμα 6.1

Βήματα που πρέπει να ακολουθήσουμε

Βήμα 1

Καθορισμός προδιαγραφών :

Βαθυπερατό FIR φίλτρο με τη μέθοδο παραθύρων τυπου (hanning)

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >50dB

Τάξη του φίλτρου N=4;

Πλάτος πλευρικου λοβού -41 dB;

Εύρος ζώνης μετάβασης Δω= 3.3/N;

Εξασθένιση στη ζώνη αποκοπής -53dB;

Βήμα 2

Υπολογισμός συντελεστών των φίλτρων :

Δεδομένα

$D\omega = \frac{3.3}{\mathrm{N}}$ =3.3/34=0.825

Τύπος Παραθύρου (hanning) $w(n) = 0.5 + 0.5\cos\frac{2\pi n}{N}$

Κρουστική απόκριση βαθυπερατού φίλτρου:

$hd(n) = 2\omega c\frac{\sin(n\omega c)}{n\omega c}$ για $n \neq 0$

$hd(n) = 2\omega c$ για $n = 0$

Οι συντελεστές υπολογίζονται απο το τύπο $h(n) = hd(n).w(n)$

Συχνότητα αποκοπής ωc

$\omega c = \omega p + D\omega / fs$

ωc=(5+0.825)=5.825/48000=0.1213

Για n=0 hd(0)=2.ωc=2*0.1213=0.242

w(0)=0.5+0.5*cos(0)=1

h(0)=h(d).w(0)=0.242

Για n=1 hd(1)=(2*0.1213/2*pi*0.1213)*sin(2*pi*0.1213)

hd(1)=sin(360*0.1213)/pi=0.219

w(1)=0.5+0.5cos(2*pi/4)

w(1)=0.5+0.5cos(360/4)=0.5

h(1)=0.219*0.5=0.10953

Για n=2 hd(2)=sin(2*360*0.1213)/pi=0.318

w(2)=0.5+0.5cos(2*360/4)=0

h(2)=0.318*0=0

Για n=3 hd(3)=sin(3*360*0.1213)/pi=0.2403

w(3)=0.5+0.5cos(3*360*0.1213)/pi=0.3955

h(2)=0.2403*0.3955=0.09503

Βήμα 3

Υπολογισμός συνάρτησης μεταφοράς

Τύπος συνάρτησης μεταφοράς

› $H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(k)z^{-k}$ (3.1)

› $H(z)=\frac{y(z)}{x(z)}=a0+a1z^{-1}+a2z^{-2}+a3z^{-3}$

› $H(z)=\frac{y(z)}{x(z)}=0.242+0.10953z^{-1}+0z^{-2}+0.09503z^{-3}$

6.2 Υπολογισμός συνάρτησης μεταφοράς IIR φίλτρου

Παράδειγμα 6.1

Βήματα που πρέπει να ακολουθήσουμε

- Από τις προδιαγραφές του ψηφιακού βρίσκουμε τις αντίστοιχες προδιαγραφές του αναλογικού βαθυπερατού φίλτρου και στη συνέχεια την συνάρτηση H(s).
- Μετατρέπουμε το αναλογικό βαθυπερατό στο αντίστοιχο αναλογικό υψυπερατό ή διέλευσης ή αποκοπής ζώνης.
- Εφαρμόζουμε τον διγραμμικό μετασχηματισμό.

Βήμα 1

Καθορισμός προδιαγραφών :

Βαθυπερατό IIR κανονικοποιημένου φίλτρου butterworth

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz

Βήμα 2

Δίνεται η αναλογική συνάρτηση(Butterworth) $H(s)=1/(s^2+\sqrt{2}s+1)$

Η κανονικοποιημένη (ψηφιακή) συχνότητα είναι: ωc=5000/48000=0.1041π

Η αντίστοιχη αναλογική συχνότητα είναι : ωc=tan(ω/2)= 0.1657

$$H(s)s=\frac{s}{0.1657}=\frac{0.1657^2}{s^2+0.1657\sqrt{2}s+0.1657^2}=\frac{0.0274}{s^2+0.1657\sqrt{2}s+0.0274}$$

Βήμα 3

Εφαρμόζουμε το διγράμμικο μετάσχηματισμό

$$s=s\frac{z-1}{z+1}$$

$$H(z)=\frac{0.0274(1+z^{-1})^2}{(1-z^{-1})^2+2343(1-z^{-2})+0.0274}$$

ΚΕΦΑΛΑΙΟ 7

Υλοποίηση ψηφιακών φίλτρων FIR

7.1.1 Βαθυπερατό φίλτρο με τη μέθοδο (window κεφ.2.4)με παράθυρο rectangular.

7.1.1.1 Χαρακτηριστικά φίλτρου rectangular.

Βαθυπερατό FIR φίλτρο με τη μέθοδο παραθύρων τυπου (rectangular)

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >21dB

Τάξη του φίλτρου N=34;

Πλάτος πλευρικου λοβού -13 dB;

Εύρος ζώνης μετάβασης Δω= 0.9/N;

Εξασθένιση στη ζώνη αποκοπής -21dB;

7.1.1.2 Υπολογισμός συνάρτησης μεταφοράς FIR φίλτρου.

Για τον υπολογισμό της συνάρτης μεταφοράς χρησιμοποιούμε το πιο κάτω πρόγραμμα στο MATLAB:

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς βαθυπερατού φίλτρου (rectangular):

```
n=34;
fp=5000;
fs=48000;
fn=2*fp/fs;
window=boxcar(n+1);
b=fir1(n,fn,window);
[H W]=freqz(b,1,128);
Hz=tf(b,1,z)
```

Τύπος συνάρτησης μεταφοράς

› $H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(k)z^{-k}$ (3.1)

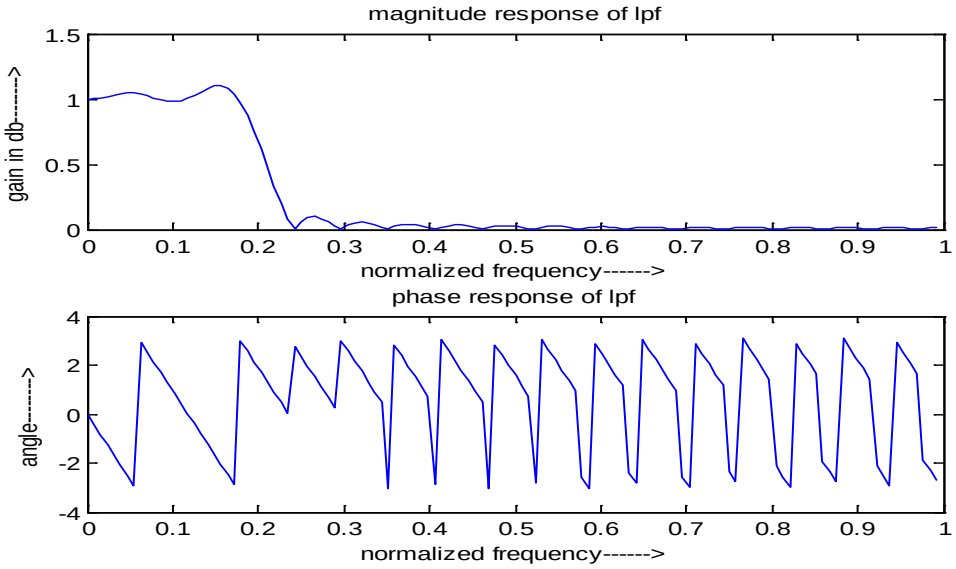
› $H(z)=\frac{y(z)}{x(z)}=a0+a1z^{-1}+a2z^{-2}+a3z^{-3}.....+aNz^{-N}$

Transfer function:

-0.01895 z^{34} - 0.01759 z^{33} - 0.008291 z^{32} + 0.006008 z^{31} + 0.01983 z^{30} + 0.02708 z^{29} + 0.02344 z^{28} + 0.008412 z^{27} - 0.01382 z^{26} - 0.03518 z^{25} - 0.04603 z^{24} - 0.0383 z^{23} - 0.008484 z^{22} + 0.04062 z^{21} + 0.1001 z^{20} + 0.157 z^{19} + 0.1978 z^{18} + 0.2127 z^{17}+ 0.1978 z^{16} + 0.157 z^{15} + 0.1001

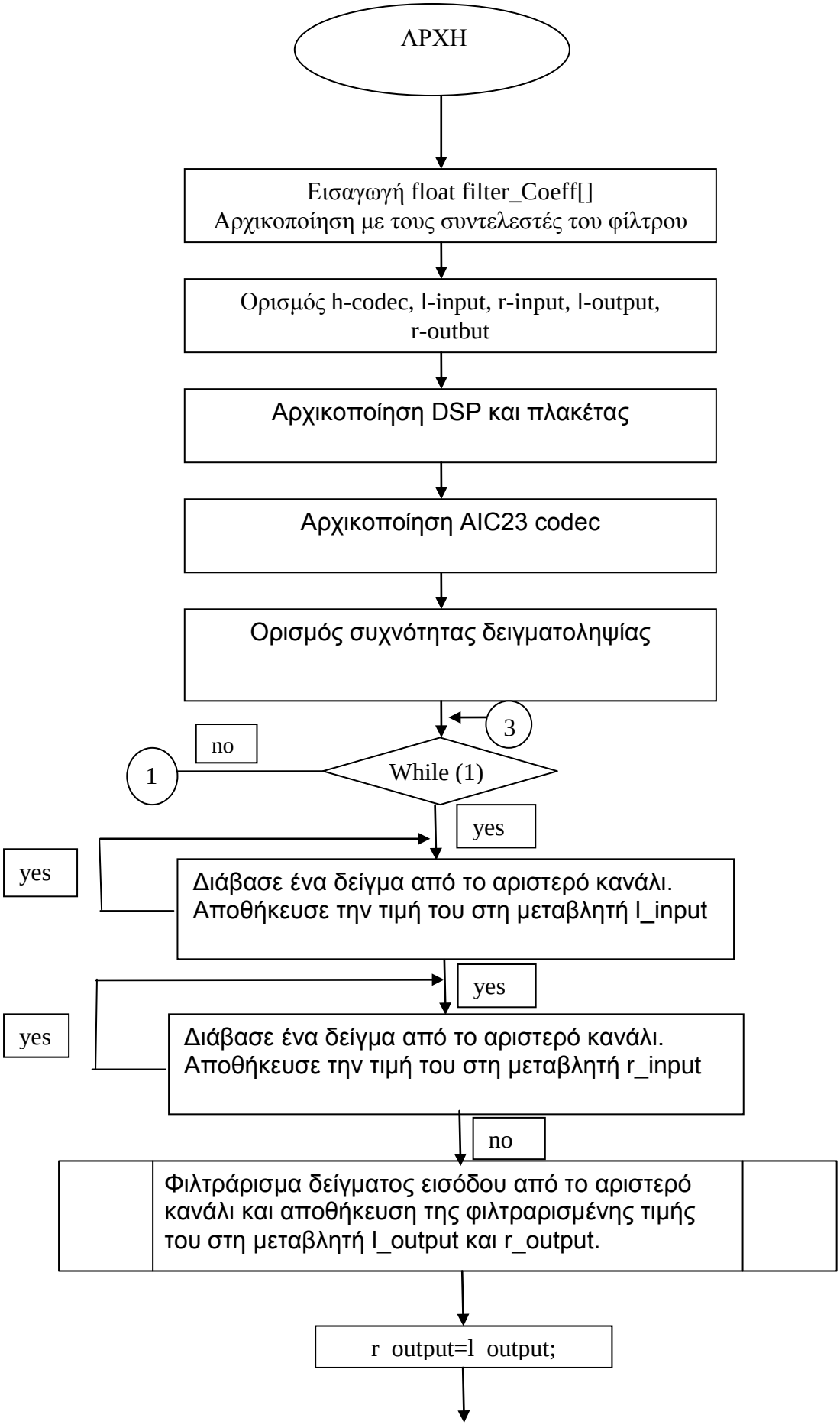
$z^{14} + 0.04062\ z^{13} - 0.008484\ z^{12} - 0.0383\ z^{11} - 0.04603\ z^{10} - 0.03518\ z^{9} - 0.01382\ z^{8} + 0.008412\ z^{7} + 0.02344\ z^{6} + 0.02708\ z^{5} + 0.01983\ z^{4} + 0.006008\ z^{3} - 0.008291\ z^{2} - 0.01759\ z - 0.01895$

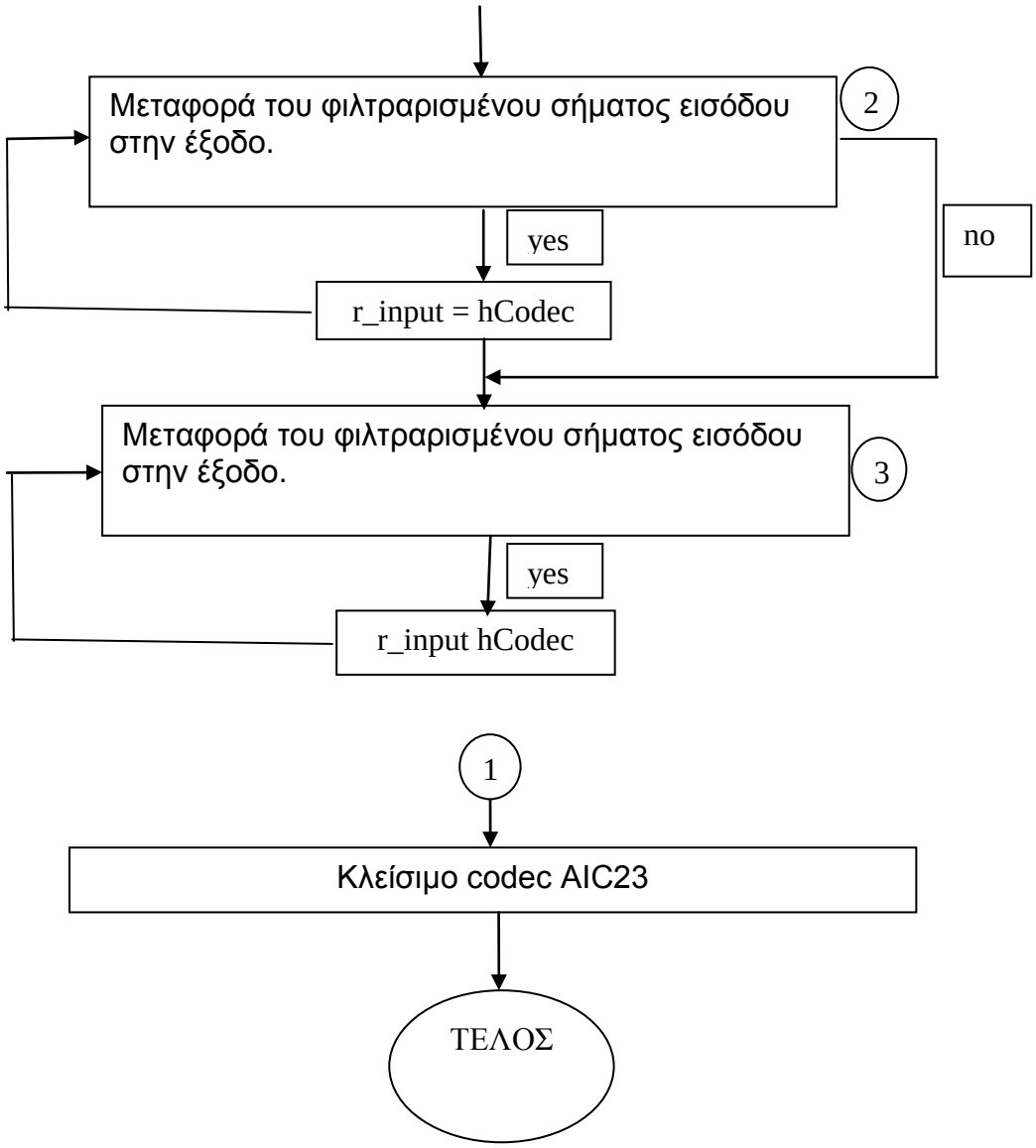
7.1.1.3 Block διάγραμμα βαθυπερατού φίλτρου (rectangular).

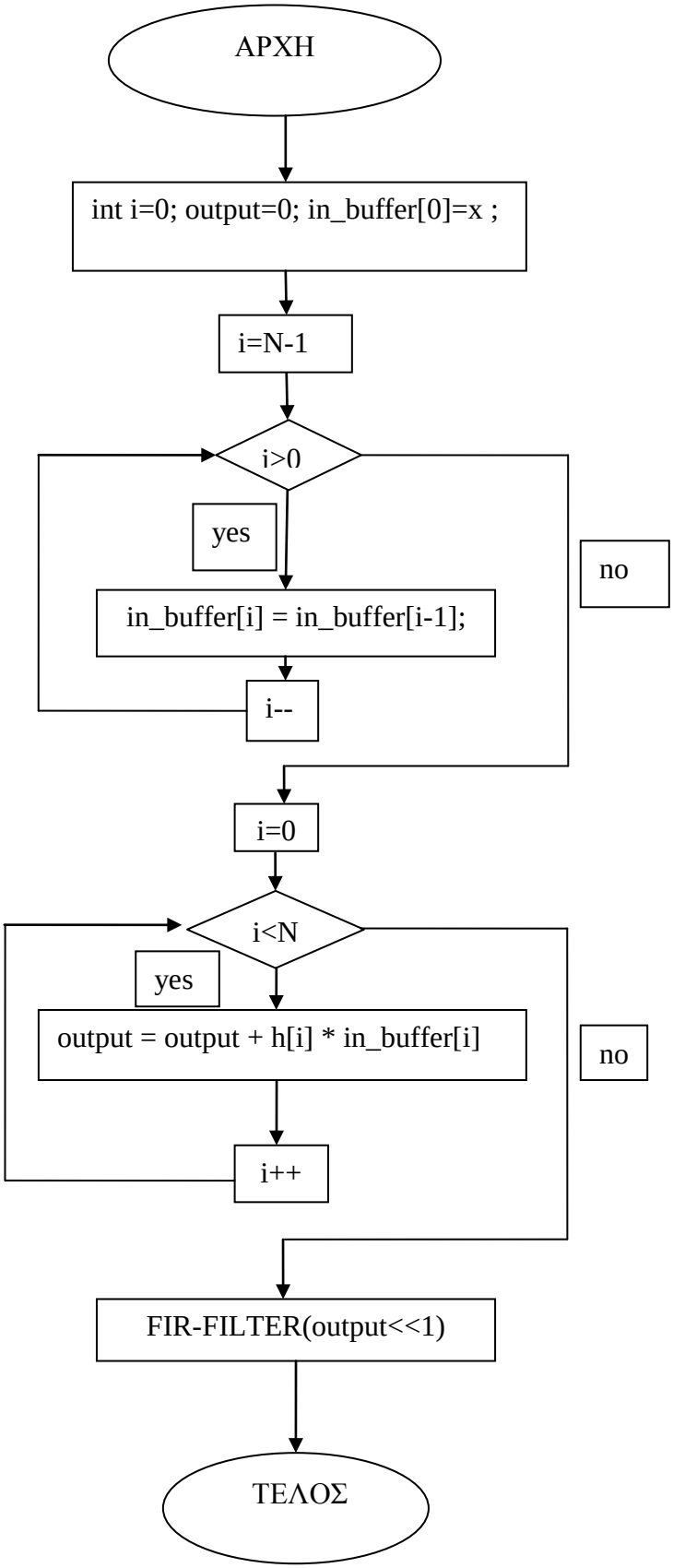


7.1.1.4 Υλοποίηση του αλγόριθμου του φίλτρου

Πιο κάτω φένεται το διάγραμμα ροής σε κώδικα C της (Texas Instruments)για FIR-FILTER. Το διάγραμμα ροής και ο κώδικας είναι ο ίδιος για όλα τα είδη FIR-FILTER,αυτο που αλλάζει κάθε φορά ανάλογα με τον τύπο του φίλτρου είναι οι συντελεστές των φίλτρων (filter coefficients)και η επιθυμητή τάξη.


ΑΡΧΗ
Εισαγωγή float filter_Coeff[]
Αρχικοποίηση με τους συντελεστές του φίλτρου
Ορισμός h-codec, l-input, r-input, l-output, r-outbut
Αρχικοποίηση DSP και πλακέτας
Αρχικοποίηση AIC23 codec
Ορισμός συχνότητας δειγματοληψίας
3
no
1
While (1)
yes
yes
Διάβασε ένα δείγμα από το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή l_input
yes
yes
Διάβασε ένα δείγμα από το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input
no
Φιλτράρισμα δείγματος εισόδου από το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητή l_output και r_output.
r_output=l_output;


Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο.
2
yes
no
r_input = hCodec
Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο.
3
yes
r_input hCodec
1
Κλείσιμο codec AIC23
ΤΕΛΟΣ


ΑΡΧΗ
int i=0; output=0; in_buffer[0]=x ;
i=N-1
i>0
yes
no
in_buffer[i] = in_buffer[i-1];
i--
i=0
i<N
yes
output = output + h[i] * in_buffer[i]
no
i++
FIR-FILTER(output<<1)
ΤΕΛΟΣ

7.1.2 Υψηπερατό φίλτρο με τη μέθοδο (window κεφ.2.4)με παράθυρο (rectangular).

7.1.2.1 Χαρακτηριστικά φίλτρου(rectangular).

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >21dB

Τάξη του φίλτρου N=34;

Πλάτος πλευρικου λοβού-13dB;

Εύρος ζώνης μετάβασης Δω= 0.9/N;

Εξασθένιση στη ζώνη αποκοπής -21dB;

7.1.2.2 Υπολογισμός συνάρτησης μεταφοράς FIR φίλτρου.

Υπολογισμός συνάρτησης μεταφοράς

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς υψηπερατού φίλτρου(rectangular):

```
n=34;
fp=7000;
fs=48000;
fn=2*fp/fs;
window=boxcar(n+1);
b=fir1(n,fn,'high',window);
[H W]=freqz(b,1,128);
Hz=tf(b,1,z)
```

Τύπος συνάρτησης μεταφοράς

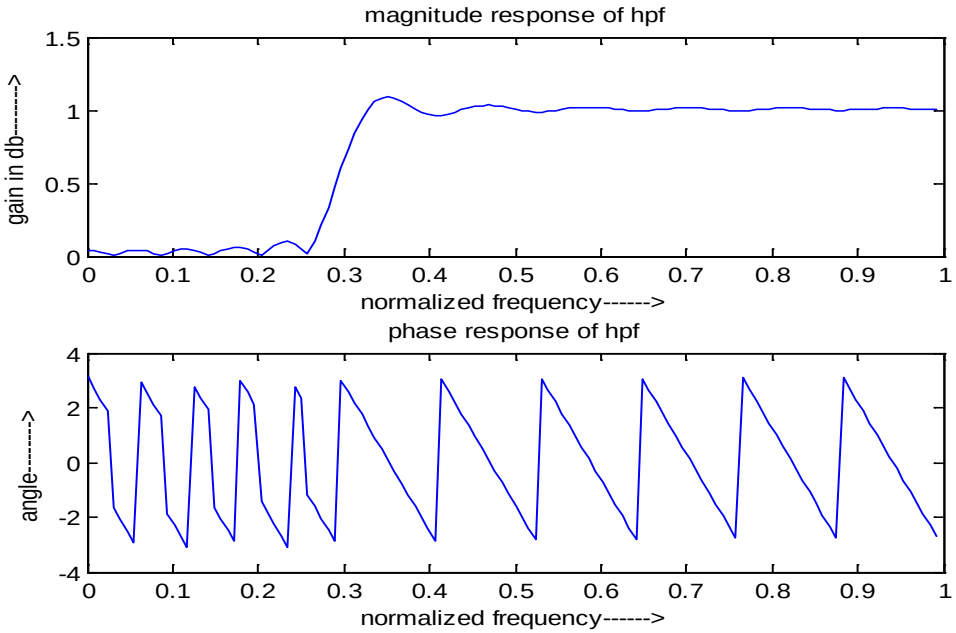
› $H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(k)z^{-k}$ (3.1)

› $H(z)=\frac{y(z)}{x(z)}=a0+a1z^{-1}+a2z^{-2}+a3z^{-3}.....+aNz^{-N}$

Transfer function:

$-0.002461\ z^{34} - 0.01735^{33} - 0.01974\ z^{32} - 0.005925\ z^{31} + 0.01501\ z^{30} + 0.02671\ z^{29} + 0.01774\ z^{28} - 0.008295\ z^{27} - 0.0329\ z^{26} - 0.03469\ z^{25} - 0.005976\ z^{24} + 0.03777\ z^{23} + 0.06355\ z^{22} + 0.04006\ z^{21} - 0.04088\ z^{20} - 0.1548\ z^{19} - 0.2543\ z^{18} + 0.7132\ z^{17} - 0.2543\ z^{16} - 0.1548\ z^{15} - 0.04088\ z^{14} + 0.04006\ z^{13} + 0.06355\ z^{12} + 0.03777\ z^{11} - 0.005976\ z^{10} - 0.03469\ z^{9} - 0.0329\ z^{8} - 0.008295\ z^{7} + 0.01774\ z^{6} + 0.02671\ z^{5} + 0.01501\ z^{4} - 0.005925\ z^{3} - 0.01974\ z^{2} - 0.01735\ z - 0.002461$

7.1.2.3 Block διάγραμμα υψηπερατού φίλτρου (rectangular).



7.1.3.4 Υλοποίηση του αλγόριθου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 34  //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[] ={
1.893444e-002,3.105706e-002,2.498957e-002,1.798033e-017,
-3.094282e-002,-4.781547e-002,-3.656879e-002,-7.817535e-018,
4.164928e-002,6.211412e-002,4.598363e-002,-6.254028e-018,
-4.939828e-002,-7.172320e-002,-5.175509e-002,0,
5.295809e-002,7.510836e-002,5.295809e-002,0,
-5.175509e-002,-7.172320e-002,-4.939828e-002,-6.254028e-018,
4.598363e-002,6.211412e-002,4.164928e-002,-7.817535e-018,
-3.656879e-002,-4.781547e-002,-3.094282e-002,1.798033e-017,
2.498957e-002,3.105706e-002,1.893444e-002};
```

```
static short in_buffer[N];  /*Πίνκας που έχει καθολική εμβέλια μέσα στο πρόγραμμα
                              και έχει οριστεί ώς static ετσι ώστε να διατηρεί τιμές που
                              αποθηκεύονται σε αυτόν σε προηγούμενες επαναλήψεις
                              του προγράμματος*/
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληψίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
```

```
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος.*/
Ο αλγόριθμος αυτός υλοποιεί το άθροισμα
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* Πάντα κάθε νέο δείγμα εισόδου, πρίν τη διαδικασία φιλτραρισματος
                     τοποθετείται στην πρώτη θέση του πίακα in_buffer (in_buffer[0]) */
for(i=(N-1);i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* Μετακίνηση όλων των δειγμάτων κατα μία θέση δεξιά */
for(i=0;i<(N);i++)
output = output + h[i] * in_buffer[i];/*Υλοποιηση του παραπάνω αθροισματος */
return(output<<1);
```

7.1.3 Διέλευσης ζώνης φίλτρο με τη μέθοδο (window κεφ.2.4)με παράθυρο rectangular.

7.1.3.1.Χαρακτηριστικά φίλτρου rectangular.

Διέλευσής ζώνης FIR φίλτρο με τη μέθοδο παραθύρων τυπου (rectangular)

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα αποκοπής ωs στη ζώνη αποκοπής :ωs=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >21dB

Τάξη του φίλτρου N=34;

Πλάτος πλευρικου λοβού -13 dB;

Εύρος ζώνης μετάβασης Δω= 0.9/N;

Εξασθένιση στη ζώνη αποκοπής -21dB;

7.1.3.2 Υπολογισμός συνάρτησης μεταφοράς FIR φίλτρου.

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς διέλευσης ζώνης φίλτρου(rectangular):

```
n=34;
fp=5000;
fq=7000;
fs=48000;
fl=2*fp/fs;
fu=2*fq/fs;
bw=[fl fu];
window=boxcar(n+1);
b=fir1(n,bw,'bandpass',window);
Hz=tf(b,1,z)
```

Τύπος συνάρτησης μεταφοράς

› $H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(k)z^{-k}$ (3.1)

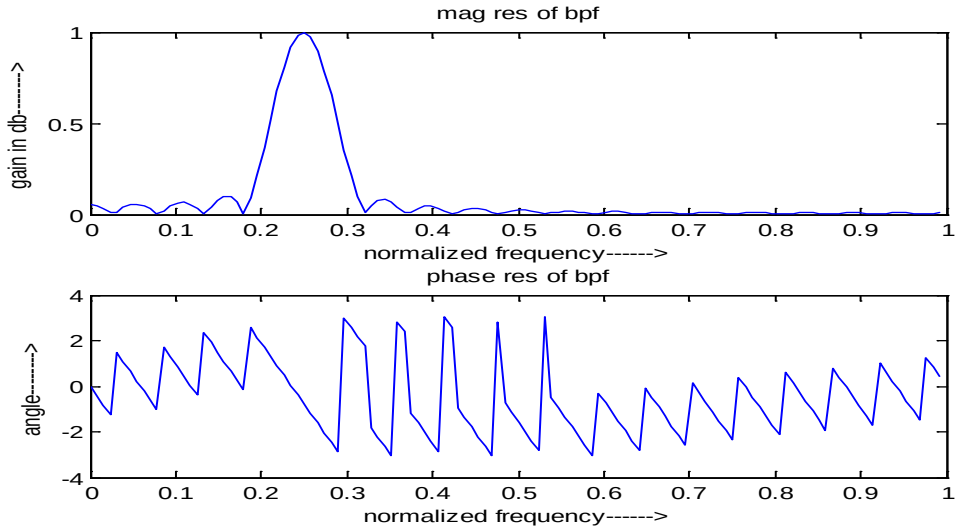
› $H(z)=\frac{y(z)}{x(z)}=a0+a1z^{-1}+a2z^{-2}+a3z^{-3}.....+aNz^{-N}$

Transfer function:

0.01893 z^{34} + 0.03106 z^{33} + 0.02499 z^{32} + 1.798e-017 z^{31} - 0.03094 z^{30} - 0.04782 z^{29}- 0.03657 z^{28} - 7.818e-018 z^{27} + 0.04165 z^{26} + 0.06211 z^{25} + 0.04598z^{24} - 6.254e-018 z^{23} - 0.0494 z^{22} - 0.07172 z^{21} - 0.05176 z^{20} + 0.05296 z^{18} + 0.07511 z^{17} + 0.05296 z^{16}- 0.05176 z^{14} - 0.07172 z^{13} -

$0.0494\ z^{12} - 6.254e\text{-}018\ z^{11} + 0.04598\ z^{10} + 0.06211\ z^{9} + 0.04165\ z^{8} - 7.818e\text{-}018\ z^{7} - 0.03657\ z^{6} - 0.04782\ z^{5} - 0.03094\ z^{4} + 1.798e\text{-}017\ z^{3} + 0.02499\ z^{2} + 0.03106\ z + 0.01893$

7.1.3.3 Block διάγραμμα Διέλευσης ζώνης φίλτρου (rectangular).



7.1.3.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 34  //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[] ={
1.893444e-002,3.105706e-002,2.498957e-002,1.798033e-017,
-3.094282e-002,-4.781547e-002,-3.656879e-002,-7.817535e-018,
4.164928e-002,6.211412e-002,4.598363e-002,-6.254028e-018,
-4.939828e-002,-7.172320e-002,-5.175509e-002,0,
5.295809e-002,7.510836e-002,5.295809e-002,0,
-5.175509e-002,-7.172320e-002,-4.939828e-002,-6.254028e-018,
4.598363e-002,6.211412e-002,4.164928e-002,-7.817535e-018,
-3.656879e-002,-4.781547e-002,-3.094282e-002,1.798033e-017,
2.498957e-002,3.105706e-002,1.893444e-002};
```

```
static short in_buffer[N];  /*Πίνκας που έχει καθολική εμβέλια μέσα στο πρόγραμμα
                              και έχει οριστεί ώς static ετσι ώστε να διατηρεί τιμές που
                              αποθηκεύονται σε αυτόν σε προηγούμενες επαναλήψεις
                              του προγράμματος*/
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
```

```
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος.*/
Ο αλγόριθμος αυτός υλοποιεί το άθροισμα
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* Πάντα κάθε νέο δείγμα εισόδου, πρίν τη διαδικασία φιλτραρισματος
                     τοποθετείται στην πρώτη θέση του πίακα in_buffer (in_buffer[0]) */
for(i=(N-1);i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* Μετακίνηση όλων των δειγμάτων κατα μία θέση δεξιά */
for(i=0;i<(N);i++)
output = output + h[i] * in_buffer[i];/*Υλοποιηση του παραπάνω αθροισματος */
return(output<<1); }
```

7.1.4 Αποκοπής ζώνης φίλτρο με τη μέθοδο (window κεφ.2.4)με παράθυρο rectangular.

7.1.4.1.Χαρακτηριστικά φίλτρου rectangular.

Διέλευσής ζώνης FIR φίλτρο με τη μέθοδο παραθύρων τυπου (rectangular)

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα αποκοπής ωs στη ζώνη αποκοπής :ωs=12KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >21dB

Τάξη του φίλτρου N=34;

Πλάτος πλευρικου λοβού -13 dB;

Εύρος ζώνης μετάβασης Δω= 0.9/N;

Εξασθένιση στη ζώνη αποκοπής -21dB;

7.1.4.2 Υπολογισμός συνάρτησης μεταφοράς FIR φίλτρου.

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς αποκοπής ζώνης φίλτρου(rectangular):

```
n=34;
fp=5000;
fq=12000;
fs=48000;
fl=2*fp/fs;
fu=2*fq/fs;
bw=[fl fu];
window=boxcar(n+1);
b=fir1(n,bw,'stop',window);
Hz=tf(b,1,z)
```

Τύπος συνάρτησης μεταφοράς

› $H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(k)z^{-k}$ (3.1)

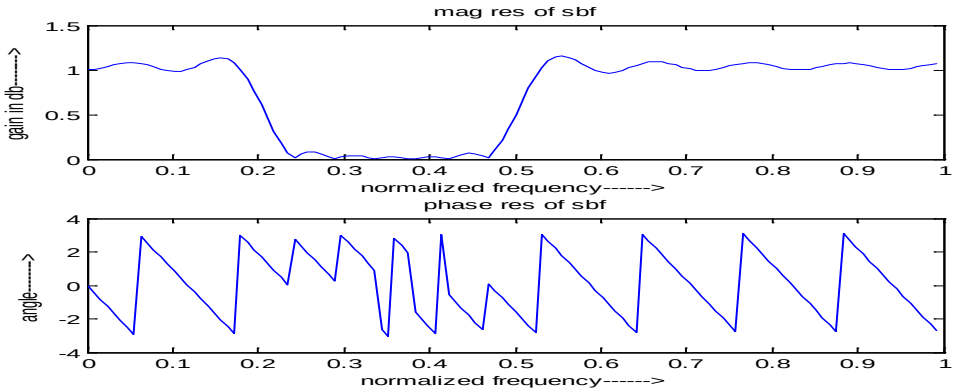
› $H(z)=\frac{y(z)}{x(z)}=a0+a1z^{-1}+a2z^{-2}+a3z^{-3}.....+aNz^{-N}$

Transfer function:

$-0.03877\ z^{34} - 0.01791\ z^{33} + 0.01362\ z^{32} + 0.006118\ z^{31} - 0.005261\ z^{30} + 0.02758\ z^{29} + 0.05396\ z^{28} + 0.008566\ z^{27} - 0.05085\ z^{26} - 0.03583\ z^{\wedge 25} + 0.0004045\ z^{\wedge 24} - 0.039\ z^{\wedge 23} - 0.07483\ z^{22} + 0.04137\ z^{21} + 0.2122\ z^{20} + 0.1598\ z^{19} - 0.1295\ z^{18} + 0.7365\ z^{17} - 0.1295\ z^{16} + 0.1598\ z^{15} + 0.2122\ z^{14} + 0.04137\ z^{13} - 0.07483\ z^{12} - 0.039\ z^{11} + 0.0004045\ z^{10} - 0.03583\ z^{9} - 0.05085\ z^{8} +$

$0.008566\ z^7 + 0.05396\ z^6 + 0.02758\ z^5 - 0.005261\ z^4 + 0.006118\ z^3 + 0.01362\ z^2 - 0.01791\ z - 0.03877$

7.1.4.3 Block διάγραμμα αποκοπής ζώνης φίλτρου (rectangular).



7.1.4.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 34  //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[] ={
1-3.876961e-002,-1.791359e-002,1.362036e-002,6.118434e-003,
-5.260864e-003,2.757978e-002,5.395668e-002,8.565807e-003,
-5.084547e-002,-3.582719e-002,4.044837e-004,-3.900370e-002,
-7.483119e-002,4.136967e-002,2.122407e-001,1.598401e-001,
-1.294833e-001,7.364777e-001,-1.294833e-001,1.598401e-001,
2.122407e-001,4.136967e-002,-7.483119e-002,-3.900370e-002,
4.044837e-004,-3.582719e-002,-5.084547e-002,8.565807e-003,
5.395668e-002,2.757978e-002,-5.260864e-003,6.118434e-003,
1.362036e-002,-1.791359e-002,-3.876961e-002};
static short in_buffer[N];  /*Πίνκας που έχει καθολική εμβέλια μέσα στο πρόγραμμα
                              και έχει οριστεί ώς static ετσι ώστε να διατηρεί τιμές που
                              αποθηκεύονται σε αυτόν σε προηγούμενες επαναλήψεις
                              του προγράμματος*/
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη
βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του
codec AIC23*/
DSK6713_AIC23_Config config = {\
```

```
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control
*/\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου
με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του
φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα
δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή
l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
```

```
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος.*/
Ο αλγόριθμος αυτός υλοποιεί το άθροισμα
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* Πάντα κάθε νέο δείγμα εισόδου, πρίν τη διαδικασία φιλτραρισματος
                     τοποθετείται στην πρώτη θέση του πίακα in_buffer (in_buffer[0]) */
for(i=(N-1);i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* Μετακίνηση όλων των δειγμάτων κατα μία θέση δεξιά */
for(i=0;i<(N);i++)
output = output + h[i] * in_buffer[i];/*Υλοποιηση του παραπάνω αθροισματος */
return(output<<1);
}
```

7.2.1 Βαθυπερατό φίλτρο με τη μέθοδο (window κεφ.2.4)με παράθυρο hamming.

7.2.1.1 Χαρακτηριστικά φίλτρου hamming.

Βαθυπερατό FIR φίλτρο με τη μέθοδο παραθύρων τυπου (hamming)

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >44dB

Τάξη του φίλτρου N=34;

Πλάτος πλευρικου λοβού -31 dB;

Εύρος ζώνης μετάβασης Δω= 3.1/N;

Εξασθένιση στη ζώνη αποκοπής -44dB;

7.2.1.2 Υπολογισμός συνάρτησης μεταφοράς FIR φίλτρου.

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς βαθυπερατού φιλτρου(hamming):

```
n=34;
fp=5000;
fs=48000;
fn=2*fp/fs;
window=hamming(n+1);
b=fir1(n,fn,window);
Hz=tf(b,1,z)
```

Τύπος συνάρτησης μεταφοράς

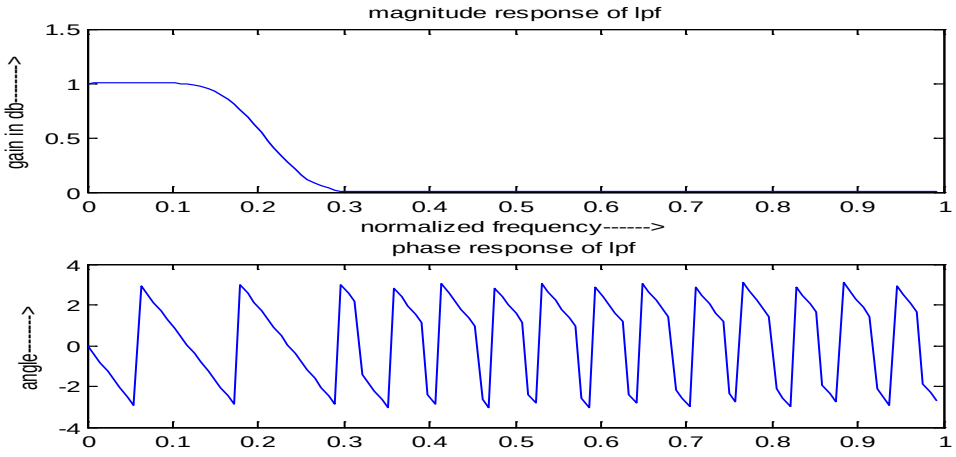
› $H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(k)z^{-k}$ (3.1)

› $H(z)=\frac{y(z)}{x(z)}=a0+a1z^{-1}+a2z^{-2}+a3z^{-3}.....+aNz^{-N}$

Transfer function:

$-0.001488\ z^{34} - 0.001516\ z^{33} - 0.0009037\ z^{32} + 0.0008779\ z^{31} + 0.003894\ z^{30} + 0.006984\ z^{29} + 0.007705\ z^{28} + 0.003418\ z^{27} - 0.006747\ z^{26} - 0.02011\ z^{25} - 0.03008\ z^{24} - 0.028\ z^{23} - 0.006804\ z^{22} + 0.03508\ z^{21} + 0.09145\ z^{20} + 0.1492\ z^{19} + 0.1926\ z^{18} + 0.2087\ z^{17} + 0.1926\ z^{16} + 0.1492\ z^{15} + 0.09145\ z^{14} + 0.03508\ z^{13} - 0.006804\ z^{12} - 0.028\ z^{11} - 0.03008\ z^{10} - 0.02011\ z^{9} - 0.006747\ z^{8} + 0.003418\ z^{7} + 0.007705\ z^{6} + 0.006984\ z^{5} + 0.003894\ z^{4} + 0.0008779\ z^{3} - 0.0009037\ z^{2} - 0.001516\ z - 0.001488$

7.2.1.4 Block διάγραμμα βαθυπερατού φίλτρου (hamming).



7.2.1.4 Υλοποίηση του αλγόριθου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 34  //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[] ={
-1.487986e-003,-1.516193e-003,-9.036624e-004,8.779151e-004,
3.893713e-003,6.984152e-003,7.704717e-003,3.418271e-003,
-6.747282e-003,-2.010869e-002,-3.007870e-002,-2.800286e-002,
-6.803802e-003,3.507958e-002,9.144915e-002,1.492446e-001,
1.926289e-001,2.087363e-001,1.926289e-001,1.492446e-001,
9.144915e-002,3.507958e-002,-6.803802e-003,-2.800286e-002,
-3.007870e-002,-2.010869e-002,-6.747282e-003,3.418271e-003,
7.704717e-003,6.984152e-003,3.893713e-003,8.779151e-004,
-9.036624e-004,-1.516193e-003,-1.487986e-003};
static short in_buffer[N];  /*Πίνκας που έχει καθολική εμβέλια μέσα στο πρόγραμμα
                              και έχει οριστεί ώς static ετσι ώστε να διατηρεί τιμές που
                              αποθηκεύονται σε αυτόν σε προηγούμενες επαναλήψεις
                              του προγράμματος*/
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control
*/\
```

```
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
```

```
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος.*/
Ο αλγόριθμος αυτός υλοποιεί το άθροισμα
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* Πάντα κάθε νέο δείγμα εισόδου, πρίν τη διαδικασία φιλτραρισματος
                     τοποθετείται στην πρώτη θέση του πίακα in_buffer (in_buffer[0]) */
for(i=(N-1);i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* Μετακίνηση όλων των δειγμάτων κατα μία θέση δεξιά */
for(i=0;i<(N);i++)
output = output + h[i] * in_buffer[i];/*Υλοποιηση του παραπάνω αθροισματος */
return(output<<1);
}
```

7.2.2 Υψηπερατό φίλτρο με τη μέθοδο (window κεφ.2.4)με παράθυρο hamming.

7.2.2.1Χαρακτηριστικά φίλτρου hamming.

Υψηπερατό FIR φίλτρο με τη μέθοδο παραθύρων τυπου (hamming)

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >44dB

Τάξη του φίλτρου N=34;

Πλάτος πλευρικου λοβού -31 dB;

Εύρος ζώνης μετάβασης Δω= 3.1/N;

Εξασθένιση στη ζώνη αποκοπής -44dB;

7.2.2.2 Υπολογισμός συνάρτησης μεταφοράς FIR φίλτρου.

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς υψηπερατού φίλτρου (hamming):

```
n=34;
fp=7000;
fs=48000;
fn=2*fp/fs;
window=hamming(n+1);
b=fir1(n,fn,'high',window);
```

Hz=tf(b,1,z)

Τύπος συνάρτησης μεταφοράς

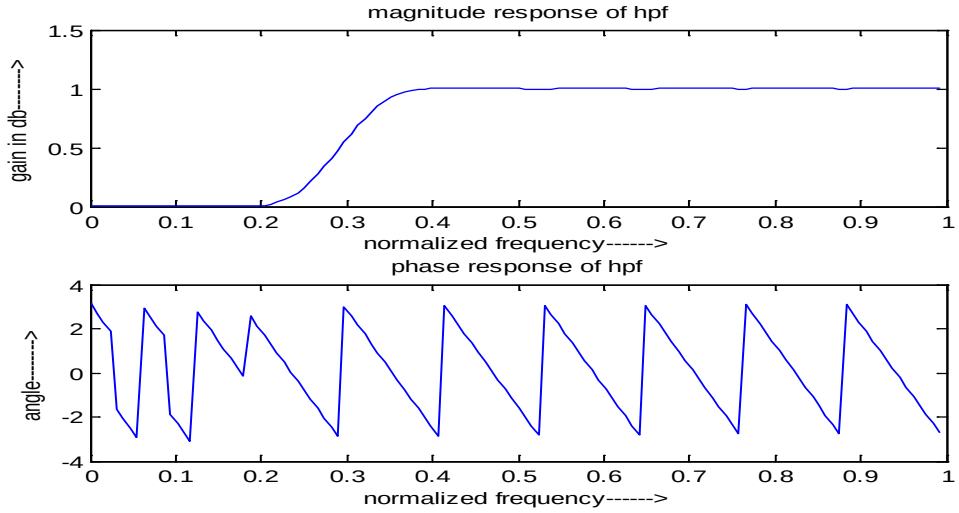
› $H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(k)z^{-k}$ (3.1)

› $H(z)=\frac{y(z)}{x(z)}=a0+a1z^{-1}+a2z^{-2}+a3z^{-3}.....+aNz^{-N}$

Transfer function:

$-0.0001956\ z^{34} - 0.001514\ z^{33} - 0.002179\ z^{32} - 0.0008767\ z^{31} + 0.002983\ z^{30} + 0.006974\ z^{29} + 0.005904\ z^{28} - 0.003413\ z^{27} - 0.01627\ z^{26} - 0.02008\ z^{25} - 0.003954\ z^{24} + 0.02796\ z^{23} + 0.05161\ z^{22} + 0.03503\ z^{21} - 0.03783\ z^{20} - 0.149\ z^{19} - 0.2507\ z^{18} + 0.7087\ z^{17} - 0.2507\ z^{16} - 0.149\ z^{15} - 0.03783\ z^{14} + 0.03503\ z^{13} + 0.05161\ z^{12} + 0.02796\ z^{11} - 0.003954\ z^{10} - 0.02008\ z^{9} - 0.01627\ z^{8} - 0.003413\ z^{7} + 0.005904\ z^{6} + 0.006974\ z^{5} + 0.002983\ z^{4} - 0.0008767\ z^{3} - 0.002179\ z^{2} - 0.001514\ z - 0.0001956$

7.2.2.3 Block διάγραμμα υψηπερατού φίλτρου (hamming).



7.2.2.4 Υλοποίηση του αλγόριθου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 34  //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[] ={
-1.956162e-004,-1.514019e-003,-2.178506e-003,-8.766563e-004,
2.983467e-003,6.974137e-003,5.903560e-003,-3.413370e-003,
-1.626602e-002,-2.007985e-002,-3.954258e-003,2.796271e-002,
```

```
5.160590e-002,3.502928e-002,-3.782516e-002,-1.490306e-001,
-2.506789e-001,7.086857e-001,-2.506789e-001,-1.490306e-001,
-3.782516e-002,3.502928e-002,5.160590e-002,2.796271e-002,
-3.954258e-003,-2.007985e-002,-1.626602e-002,-3.413370e-003,
5.903560e-003,6.974137e-003,2.983467e-003,-8.766563e-004,
-2.178506e-003,-1.514019e-003,-1.956162e-004};
static short in_buffer[N];  /*Πίνκας που έχει καθολική εμβέλια μέσα στο πρόγραμμα
                              και έχει οριστεί ώς static ετσι ώστε να διατηρεί τιμές που
                              αποθηκεύονται σε αυτόν σε προηγούμενες επαναλήψεις
                               του προγράμματος*/
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη
βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του
codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control
*/\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου
με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του
φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
```

```
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος.*/
Ο αλγόριθμος αυτός υλοποιεί το άθροισμα
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* Πάντα κάθε νέο δείγμα εισόδου, πρίν τη διαδικασία φιλτραρισματος
        τοποθετείται στην πρώτη θέση του πίακα in_buffer (in_buffer[0]) */
```

```
for(i=(N-1);i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* Μετακίνηση όλων των δειγμάτων κατα μία θέση δεξιά */
for(i=0;i<(N);i++)
output = output + h[i] * in_buffer[i];/*Υλοποιηση του παραπάνω αθροισματος */
return(output<<1);
}
```

7.2.3 Διέλευσής ζώνης φίλτρο με τη μέθοδο (window κεφ.2.4)με παράθυρο hamming.

7.2.3.1 Χαρακτηριστικά φίλτρου hamming.

Διέλευσής ζώνης FIR φίλτρο με τη μέθοδο παραθύρων τυπου (hamming)

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα αποκοπής ωs στη ζώνη αποκοπής :ωs=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >44dB

Τάξη του φίλτρου N=34;

Πλάτος πλευρικου λοβού -31 dB;

Εύρος ζώνης μετάβασης Δω= 3.1/N;

Εξασθένιση στη ζώνη αποκοπής -44dB;

7.2.3.2 Υπολογισμός συνάρτησης μεταφοράς FIR φίλτρου.

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς διέλευσης ζώνης φίλτρου (hamming):

```
n=34;
fp=5000;
fq=7000;
fs=48000;
fl=2*fp/fs;
fu=2*fq/fs;
bw=[fl fu];
window=hamming(n+1);
b=fir1(n,bw,'bandpass',window);
Hz=tf(b,1,z)
```

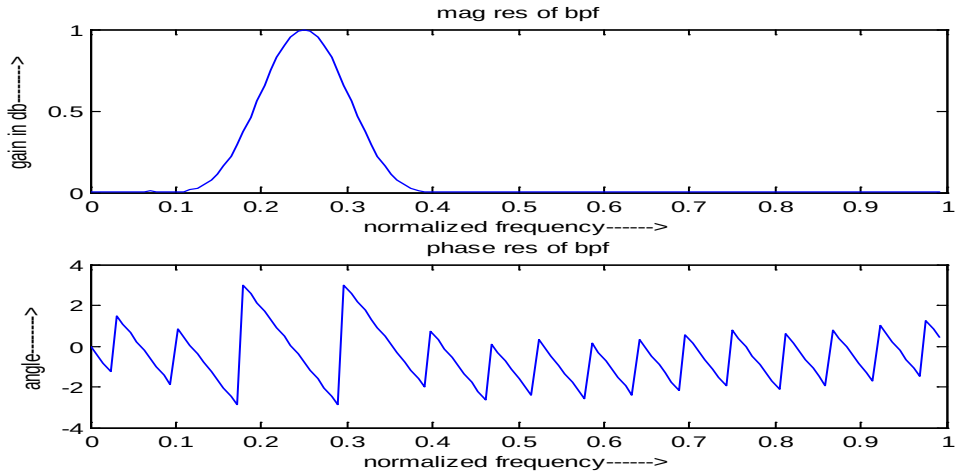
Τύπος συνάρτησης μεταφοράς

› $H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(k)z^{-k}$ (3.1)

› $H(z)=\frac{y(z)}{x(z)}=a0+a1z^{-1}+a2z^{-2}+a3z^{-3}.....+aNz^{-N}$

$0.002489\ z^{34} + 0.004482\ z^{33} + 0.00456\ z^{32} + 4.399e\text{-}018\ z^{31} - 0.01017\ z^{30} - 0.02065\ z^{29} - 0.02013\ z^{28} - 5.32e\text{-}018\ z^{27} + 0.03405\ z^{26} + 0.05945\ z^{25} + 0.05031\ z^{24} - 7.656e\text{-}018\ z^{23} - 0.06633\ z^{22} - 0.1037\ z^{21} - 0.07918\ z^{20} + 0.08634\ z^{18} + 0.1234\ z^{17} + 0.08634\ z^{16} - 0.07918\ z^{14} - 0.1037\ z^{13} - 0.06633\ z^{12} - 7.656e\text{-}018\ z^{11} + 0.05031\ z^{10} + 0.05945\ z^{9} + 0.03405\ z^{8} - 5.32e\text{-}018\ z^{7} - 0.02013\ z^{6} - 0.02065\ z^{5} - 0.01017\ z^{4} + 4.399e\text{-}018\ z^{3} + 0.00456\ z^{2} + 0.004482\ z + 0.002489$

7.2.3.3 Block διάγραμμα Διέλευσης ζώνης φίλτρου (hamming).



7.2.3.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 34  //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[] ={
2.489012e-003,4.482286e-003,4.560494e-003,4.399236e-018,
-1.017175e-002,-2.064708e-002,-2.012744e-002,-5.319556e-018,
3.405137e-002,5.944683e-002,5.031380e-002,-7.656386e-018,
-6.633323e-002,-1.037049e-001,-7.918337e-002,0,
8.633801e-002,1.234164e-001,8.633801e-002,0,
-7.918337e-002,-1.037049e-001,-6.633323e-002,-7.656386e-018,
5.031380e-002,5.944683e-002,3.405137e-002,-5.319556e-018,
-2.012744e-002,-2.064708e-002,-1.017175e-002,4.399236e-018,
4.560494e-003,4.482286e-003,2.489012e-003};
static short in_buffer[N];  /*Πίνκας που έχει καθολική εμβέλια μέσα στο πρόγραμμα
                                και έχει οριστεί ώς static ετσι ώστε να διατηρεί τιμές που
```

```
                                    αποθηκεύονται σε αυτόν σε προηγούμενες επαναλήψεις
                                    του προγράμματος*/
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control
*/\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
```

```
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος.*/
Ο αλγόριθμος αυτός υλοποιεί το άθροισμα
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* Πάντα κάθε νέο δείγμα εισόδου, πρίν τη διαδικασία φιλτραρισματος
                     τοποθετείται στην πρώτη θέση του πίακα in_buffer (in_buffer[0]) */
for(i=(N-1);i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* Μετακίνηση όλων των δειγμάτων κατα μία θέση δεξιά */
for(i=0;i<(N);i++)
output = output + h[i] * in_buffer[i];/*Υλοποιηση του παραπάνω αθροισματος */
return(output<<1);
}
```

7.2.4.1 Αποκοπής ζώνης φίλτρο με τη μέθοδο (window κεφ.2.4)με παράθυρο hamming.

7.2.4.1 Χαρακτηριστικά φίλτρου hamming.

Διέλευσής ζώνης FIR φίλτρο με τη μέθοδο παραθύρων τυπου (hamming)

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα αποκοπής ωs στη ζώνη αποκοπής :ωs=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >44dB

Τάξη του φίλτρου N=34;

Πλάτος πλευρικου λοβού -31 dB;

Εύρος ζώνης μετάβασης Δω= 3.1/N;

Εξασθένιση στη ζώνη αποκοπής -44dB;

7.2.4.2 Υπολογισμός συνάρτησης μεταφοράς FIR φίλτρου.

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς αποκοπής ζώνης φίλτρου (hamming):

```
n=34;
fp=5000;
fq=12000;
fs=48000;
fl=2*fp/fs;
fu=2*fq/fs;
bw=[fl fu];
window=hamming(n+1);
b=fir1(n,bw,'stop',window);
Hz=tf(b,1,z)
```

Τύπος συνάρτησης μεταφοράς

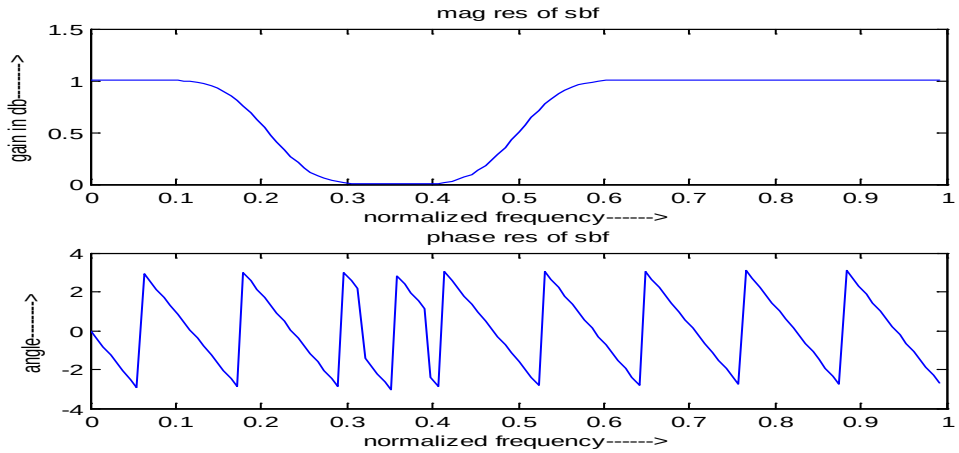
› $H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(k)z^{-k}$ (3.1)

› $H(z)=\frac{y(z)}{x(z)}=a0+a1z^{-1}+a2z^{-2}+a3z^{-3}.....+aNz^{-N}$

Transfer function:

$-0.002993\ z^{34} - 0.001518\ z^{33} + 0.00146\ z^{32} + 0.0008792\ z^{31} - 0.001016\ z^{30} + 0.006994\ z^{29} + 0.01744\ z^{28} + 0.003423\ z^{27} - 0.02441\ z^{26} - 0.02014\ z^{25} + 0.0002599\ z^{24} - 0.02804\ z^{23} - 0.05901\ z^{22} + 0.03513\ z^{21} + 0.1907\ z^{20} + 0.1495\ z^{19} - 0.124\ z^{18} + 0.7107\ z^{17} - 0.124\ z^{16} + 0.1495\ z^{15} + 0.1907\ z^{14} + 0.03513\ z^{13} - 0.05901\ z^{12} - 0.02804\ z^{11} + 0.0002599\ z^{10} - 0.02014\ z^{9} - 0.02441\ z^{8} + 0.003423\ z^{7} + 0.01744\ z^{6} + 0.006994\ z^{5} - 0.001016\ z^{4} + 0.0008792\ z^{3} + 0.00146\ z^{2} - 0.001518\ z - 0.002993$

7.2.4.3 Block διάγραμμα αποκοπής ζώνης φίλτρου (hamming).



7.2.4.4Υλοποίηση του αλγόριθου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 34  //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[] ={
-2.993094e-003,-1.518365e-003,1.459809e-003,8.791727e-004,
-1.015658e-003,6.994156e-003,1.744125e-002,3.423168e-003,
-2.441370e-002,-2.013749e-002,2.599197e-004,-2.804297e-002,
-5.901417e-002,3.512983e-002,1.907058e-001,1.494584e-001,
-1.239760e-001,7.107199e-001,-1.239760e-001,1.494584e-001,
1.907058e-001,3.512983e-002,-5.901417e-002,-2.804297e-002,
2.599197e-004,-2.013749e-002,-2.441370e-002,3.423168e-003,
1.744125e-002,6.994156e-003,-1.015658e-003,8.791727e-004,
1.459809e-003,-1.518365e-003,-2.993094e-003};
static short in_buffer[N];  /*Πίνκας που έχει καθολική εμβέλια μέσα στο πρόγραμμα
                              και έχει οριστεί ώς static ετσι ώστε να διατηρεί τιμές που
                              αποθηκεύονται σε αυτόν σε προηγούμενες επαναλήψεις
                              του προγράμματος*/
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη
βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του
codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
```

```
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control
*/\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
```

Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.

```
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
```

/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */

```
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή
l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
```

```
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος.*/
Ο αλγόριθμος αυτός υλοποιεί το άθροισμα
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* Πάντα κάθε νέο δείγμα εισόδου, πρίν τη διαδικασία φιλτραρισματος
                     τοποθετείται στην πρώτη θέση του πίακα in_buffer (in_buffer[0]) */
for(i=(N-1);i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* Μετακίνηση όλων των δειγμάτων κατα μία θέση δεξιά */
for(i=0;i<(N);i++)
output = output + h[i] * in_buffer[i];/*Υλοποιηση του παραπάνω αθροισματος */
return(output<<1);
}
```

7.3.1.1 Βαθυπερατό φίλτρο με τη μέθοδο (window κεφ.2.4)με παράθυρο hanning.

7.3.1.1 Χαρακτηριστικά φίλτρου hanning.

Βαθυπερατό FIR φίλτρο με τη μέθοδο παραθύρων τυπου (hanning)

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >53dB

Τάξη του φίλτρου N=34;

Πλάτος πλευρικου λοβού -41 dB;

Εύρος ζώνης μετάβασης Δω= 0.9/N;

Εξασθένιση στη ζώνη αποκοπής -53dB;

7.3.1.2 Υπολογισμός συνάρτησης μεταφοράς FIR φίλτρου.

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς βαθυπερατού φίλτρου (hanning):

```
n=34;
fp=5000;
fs=48000;
fn=2*fp/fs
window=hann(n+1);
b=fir1(n,fn,window);
Hz=tf(b,1,z)
```

Τύπος συνάρτησης μεταφοράς

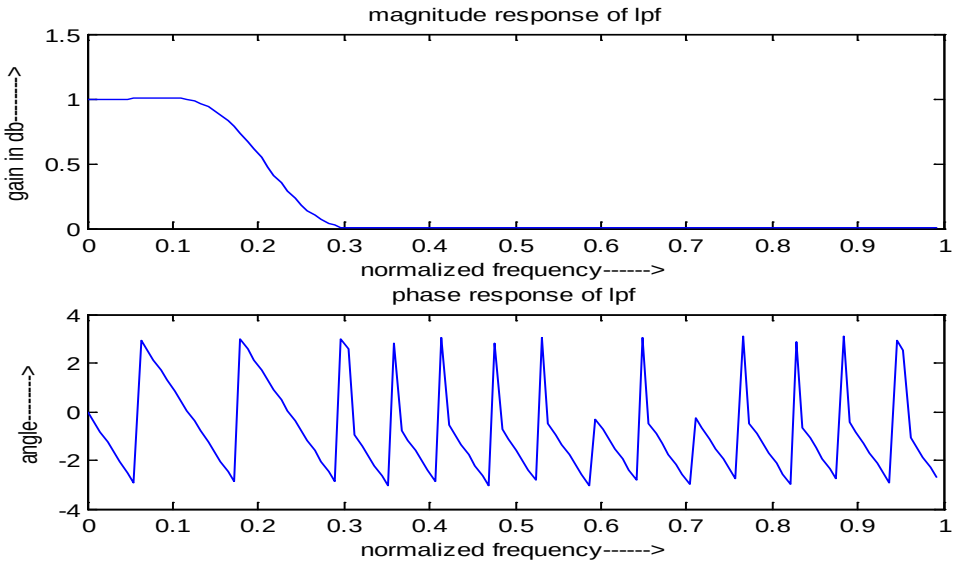
$$H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(k)z^{-k}$$

› $$H(z)=\frac{y(z)}{x(z)}=a0+a1z^{-1}+a2z^{-2}+a3z^{-3}.....+aNz^{-N}$$

Transfer function:

-0.0001467 z^{33} - 0.0002743 z^{32} + 0.0004408 z^{31} + 0.002536 z^{30} + 0.005272 z^{29} + 0.006364 z^^{28}+ 0.002993 z^{27} - 0.006145 z^{26} - 0.01882 z^{25} - 0.02872 z^{24} - 0.02713 z^{23} - 0.006661 z^{22}+ 0.03461 z^{21} + 0.09071 z^{20} + 0.1486 z^{19} + 0.1922 z^{18} + 0.2084 z^{17} + 0.1922 z^{16}+ 0.1486 z^{15} + 0.09071 z^{14} + 0.03461 z^{13} - 0.006661 z^{12} - 0.02713 z^{11} - 0.02872 z^{10}- 0.01882 z^{9} - 0.006145 z^{8} + 0.002993 z^{7} + 0.006364 z^{6} + 0.005272 z^{5} + 0.002536 z^{4}+ 0.0004408 z^{3} - 0.0002743 z^{2} - 0.0001467 z

7.3.1.3 Block διάγραμμα βαθυπερατού φίλτρου (hanning).



7.3.1.4Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 34  //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[] ={
0,-1.467237e-004,-2.742741e-004,4.408434e-004,
2.535730e-003,5.271847e-003,6.364185e-003,2.992874e-003,
-6.144813e-003,-1.882453e-002,-2.871965e-002,-2.712548e-002,
-6.660644e-003,3.460715e-002,9.071327e-002,1.485871e-001,
1.921843e-001,2.083976e-001,1.921843e-001,1.485871e-001,
9.071327e-002,3.460715e-002,-6.660644e-003,-2.712548e-002,
-2.871965e-002,-1.882453e-002,-6.144813e-003,2.992874e-003,
6.364185e-003,5.271847e-003,2.535730e-003,4.408434e-004,
-2.742741e-004,-1.467237e-004,0};
static short in_buffer[N];  /*Πίνκας που έχει καθολική εμβέλια μέσα στο πρόγραμμα
                              και έχει οριστεί ώς static ετσι ώστε να διατηρεί τιμές που
                              αποθηκεύονται σε αυτόν σε προηγούμενες επαναλήψεις
                              του προγράμματος*/
```

/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/

```
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control
*/\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
```

Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.

```
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
```

/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */

```
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
```

```
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος.*/
Ο αλγόριθμος αυτός υλοποιεί το άθροισμα
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* Πάντα κάθε νέο δείγμα εισόδου, πρίν τη διαδικασία φιλτραρισματος
                     τοποθετείται στην πρώτη θέση του πίακα in_buffer (in_buffer[0]) */
for(i=(N-1);i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* Μετακίνηση όλων των δειγμάτων κατα μία θέση δεξιά */
for(i=0;i<(N);i++)
output = output + h[i] * in_buffer[i];/*Υλοποιηση του παραπάνω αθροισματος */
return(output<<1);
}
```

7.3.2.1 Υψηπερατό φίλτρο με τη μέθοδο (window κεφ.2.4)με παράθυρο hanning.

7.3.2.1 Χαρακτηριστικά φίλτρου hanning.

Υψηπερατό FIR φίλτρο με τη μέθοδο παραθύρων τυπου (hanning)

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >53dB

Τάξη του φίλτρου N=34;

Πλάτος πλευρικου λοβού -41 dB;

Εύρος ζώνης μετάβασης Δω= 3.3/N;

Εξασθένιση στη ζώνη αποκοπής -53dB;

7.3.2.2 Υπολογισμός συνάρτησης μεταφοράς FIR φίλτρου.

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς υψηπερατού φίλτρου (hanning):

```
n=34;
fp=7000;
fs=48000;
fn=2*fp/fs;
window=hann(n+1);
b=fir1(n,fn,'high',window);
Hz=tf(b,1,z)
```

Τύπος συνάρτησης μεταφοράς

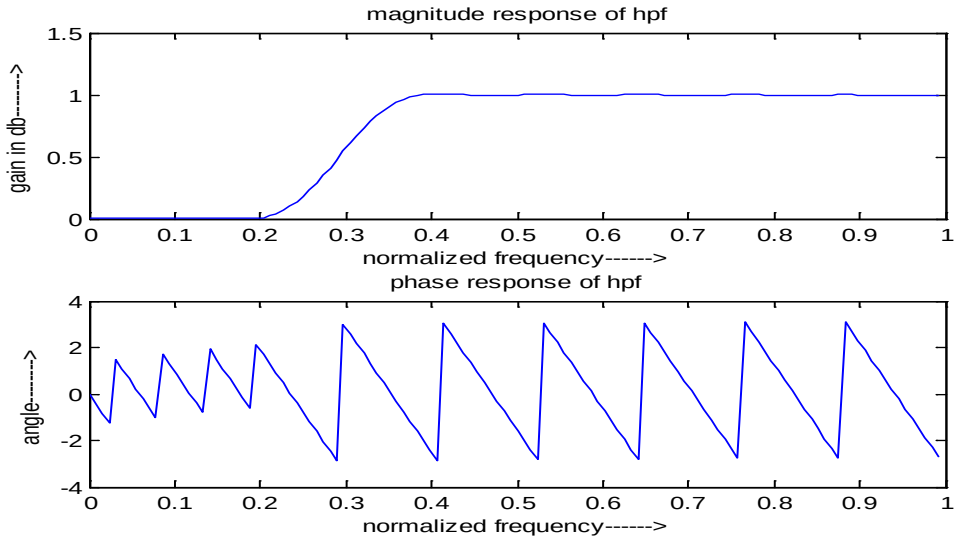
› $H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(k)z^{-k}$ (3.1)

› $H(z)=\frac{y(z)}{x(z)}=a0+a1z^{-1}+a2z^{-2}+a3z^{-3}.....+aNz^{-N}$

Transfer function:

$-0.0001467\ z^{33} - 0.0006619\ z^{32} - 0.0004407\ z^{31} + 0.001945\ z^{30} + 0.00527\ z^{29} + 0.004882\ z^{28} - 0.002992\ z^{27} - 0.01483\ z^{26} - 0.01882\ z^{25} - 0.00378\ z^{24} + 0.02712\ z^{23} + 0.05057\ z^{22} + 0.03459\ z^{21} - 0.03756\ z^{20} - 0.1485\ z^{19} - 0.2504\ z^{18} + 0.7083\ z^{17} - 0.2504\ z^{16} - 0.1485\ z^{15} - 0.03756\ z^{14} + 0.03459\ z^{13} + 0.05057\ z^{12} + 0.02712\ z^{11} - 0.00378\ z^{10} - 0.01882\ z^{9} - 0.01483\ z^{8} - 0.002992\ z^{7} + 0.004882\ z^{6} + 0.00527\ z^{5} + 0.001945\ z^{4} - 0.0004407\ z^{3} - 0.0006619\ z^{2} - 0.0001467\ z$

7.3.2.3 Block διάγραμμα υψηπερατού φίλτρου (hanning).



7.3.2.4Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 34  //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[] ={
0,-1.466713e-004,-6.619197e-004,-4.406859e-004,
1.945039e-003,5.269963e-003,4.881666e-003,-2.991804e-003,
-1.482959e-002,-1.881780e-002,-3.779662e-003,2.711579e-002,
5.057453e-002,3.459478e-002,-3.756124e-002,-1.485340e-001,
-2.503700e-001,7.082985e-001,-2.503700e-001,-1.485340e-001,
-3.756124e-002,3.459478e-002,5.057453e-002,2.711579e-002,
-3.779662e-003,-1.881780e-002,-1.482959e-002,-2.991804e-003,
4.881666e-003,5.269963e-003,1.945039e-003,-4.406859e-004,
-6.619197e-004,-1.466713e-004,0};
static short in_buffer[N];  /*Πίνκας που έχει καθολική εμβέλια μέσα στο πρόγραμμα
                              και έχει οριστεί ώς static ετσι ώστε να διατηρεί τιμές που
                              αποθηκεύονται σε αυτόν σε προηγούμενες επαναλήψεις
                              του προγράμματος*/
```

```
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
```

```
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος.*/
Ο αλγόριθμος αυτός υλοποιεί το άθροισμα
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* Πάντα κάθε νέο δείγμα εισόδου, πρίν τη διαδικασία φιλτραρισματος
                     τοποθετείται στην πρώτη θέση του πίακα in_buffer (in_buffer[0]) */
for(i=(N-1);i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* Μετακίνηση όλων των δειγμάτων κατα μία θέση δεξιά */
for(i=0;i<(N);i++)
output = output + h[i] * in_buffer[i];/*Υλοποιηση του παραπάνω αθροισματος */
return(output<<1);
}
```

7.3.3 Διέλευσής ζώνης φίλτρο με τη μέθοδο (window κεφ.2.4)με παράθυρο hanning.

7.2.3.1Χαρακτηριστικά φίλτρου hanning.

Διέλευσής ζώνης FIR φίλτρο με τη μέθοδο παραθύρων τυπου (hanning)

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα αποκοπής ωs στη ζώνη αποκοπής :ωs=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >53dB

Τάξη του φίλτρου N=34;

Πλάτος πλευρικου λοβού -41 dB;

Εύρος ζώνης μετάβασης Δω= 3.3/N;

Εξασθένιση στη ζώνη αποκοπής -53dB;

7.3.3.2 Υπολογισμός συνάρτησης μεταφοράς FIR φίλτρου.

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς διέλευσης ζώνης φίλτρου (hanning):

```
n=34;
fp=5000;
fq=7000;
fs=48000;
fl=2*fp/fs;
fu=2*fq/fs;
bw=[fl fu];
window=hann(n+1);
b=fir1(n,bw,'bandpass',window);
Hz=tf(b,1,z)
```

Τύπος συνάρτησης μεταφοράς

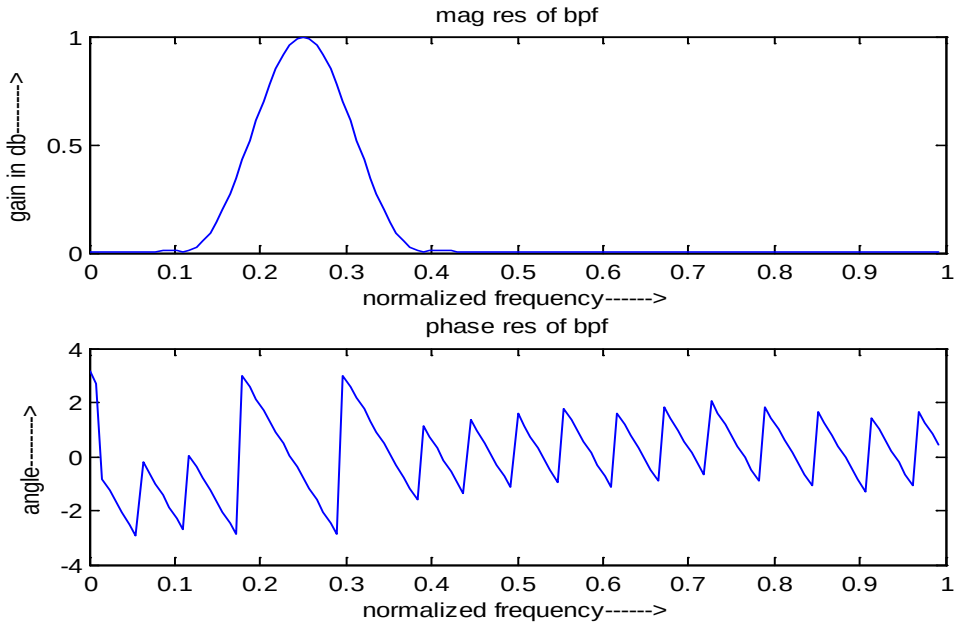
› $H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(k)z^{-k}$ (3.1)

› $H(z)=\frac{y(z)}{x(z)}=a0+a1z^{-1}+a2z^{-2}+a3z^{-3}.....+aNz^{-N}$

Transfer function:

$0.0004602\ z^{33} + 0.001469\ z^{32} + 2.344e\text{-}018\ z^{31} - 0.007028\ z^{30} - 0.01654\ z^{29} - 0.01764\ z^{28} - 4.941e\text{-}018\ z^{27} + 0.0329\ z^{26} + 0.05904\ z^{25} + 0.05097\ z^{24} - 7.869e\text{-}018\ z^{23} - 0.0689\ z^{22} - 0.1085\ z^{21} - 0.08333\ z^{20} + 0.09139\ z^{18} + 0.1307\ z^{17} + 0.09139\ z^{16} - 0.08333\ z^{14} - 0.1085\ z^{13} - 0.0689\ z^{12} - 7.869e\text{-}018\ z^{11} + 0.05097\ z^{10} + 0.05904\ z^{9} + 0.0329\ z^{8} - 4.941e\text{-}018\ z^{7} - 0.01764\ z^{6} - 0.01654\ z^{5} - 0.007028\ z^{4} + 2.344e\text{-}018\ z^{3} + 0.001469\ z^{2} + 0.0004602\ z$

7.3.3.3 Block διάγραμμα Διέλευσης ζώνης φίλτρου (hanning).



7.3.3.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 34  //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[] ={
0,4.601990e-004,1.468557e-003,2.343741e-018,
-7.028058e-003,-1.653515e-002,-1.763905e-002,-4.941488e-018,
3.290143e-002,5.904314e-002,5.096917e-002,-7.868632e-018,
-6.889633e-002,-1.085454e-001,-8.333463e-002,0,
9.139005e-002,1.307278e-001,9.139005e-002,0,
-8.333463e-002,-1.085454e-001,-6.889633e-002,-7.868632e-018,
5.096917e-002,5.904314e-002,3.290143e-002,-4.941488e-018,
-1.763905e-002,-1.653515e-002,-7.028058e-003,2.343741e-018,
1.468557e-003,4.601990e-004,0};
static short in_buffer[N];  /*Πίνκας που έχει καθολική εμβέλια μέσα στο πρόγραμμα
                              και έχει οριστεί ώς static ετσι ώστε να διατηρεί τιμές που
                              αποθηκεύονται σε αυτόν σε προηγούμενες επαναλήψεις
                              του προγράμματος*/
```

```
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
```

```
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος.*/
Ο αλγόριθμος αυτός υλοποιεί το άθροισμα
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* Πάντα κάθε νέο δείγμα εισόδου, πρίν τη διαδικασία φιλτραρισματος
                     τοποθετείται στην πρώτη θέση του πίακα in_buffer (in_buffer[0]) */
for(i=(N-1);i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* Μετακίνηση όλων των δειγμάτων κατα μία θέση δεξιά */
for(i=0;i<(N);i++)
output = output + h[i] * in_buffer[i];/*Υλοποιηση του παραπάνω αθροισματος */
return(output<<1);
}
```

7.3.4 Αποκοπής ζώνης φίλτρο με τη μέθοδο (window κεφ.2.4)με παράθυρο hanning.

7.3.4.1 Χαρακτηριστικά φίλτρου hanning.

Διέλευσής ζώνης FIR φίλτρο με τη μέθοδο παραθύρων τυπου (hanning)

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα αποκοπής ωs στη ζώνη αποκοπής :ωs=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >53dB

Τάξη του φίλτρου N=34;

Πλάτος πλευρικου λοβού -41 dB;

Εύρος ζώνης μετάβασης Δω= 3.3/N;

Εξασθένιση στη ζώνη αποκοπής -53dB;

7.3.4.2 Υπολογισμός συνάρτησης μεταφοράς FIR φίλτρου.

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς αποκοπής ζώνης φίλτρου (hanning):

```
n=34;
fp=5000;
fq=12000;
fs=48000;
fl=2*fp/fs;
fu=2*fq/fs;
bw=[fl fu];
window=hann(n+1);
b=fir1(n,bw,'stop',window);
Hz=tf(b,1,z)
```

Τύπος συνάρτησης μεταφοράς

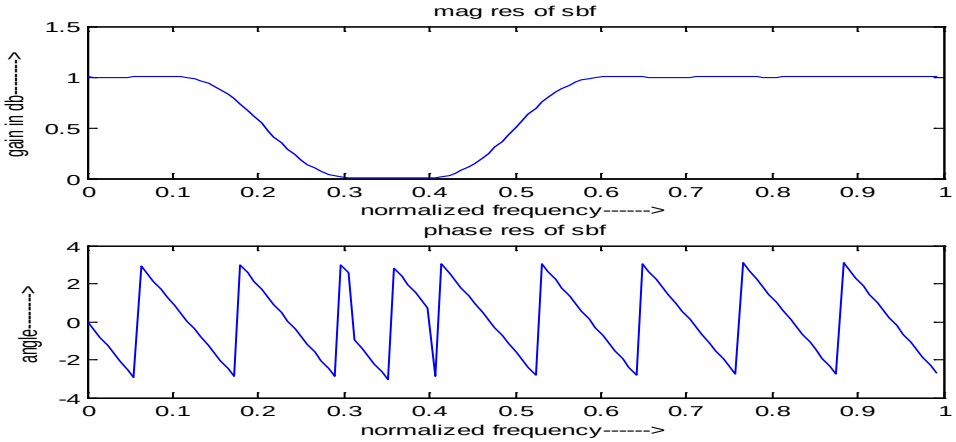
› $H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(k)z^{-k}$ (3.1)

› $H(z)=\frac{y(z)}{x(z)}=a0+a1z^{-1}+a2z^{-2}+a3z^{-3}.....+aNz^{-N}$

Transfer function:

$-0.0001467\ z^{33} + 0.0004424\ z^{32} + 0.0004409\ z^{31} - 0.0006605\ z^{30} + 0.005272\ z^{29} + 0.01439\ z^{28} + 0.002993\ z^{27} - 0.0222\ z^{26} - 0.01882\ z^{25} + 0.0002478\ z^{24} - 0.02713\ z^{23} - 0.05769\ z^{22} + 0.03461\ z^{21} + 0.1889\ z^{20} + 0.1486\ z^{19} - 0.1235\ z^{18} + 0.7086\ z^{17} - 0.1235\ z^{16} + 0.1486\ z^{15} + 0.1889\ z^{14} + 0.03461\ z^{13} - 0.05769\ z^{12} - 0.02713\ z^{11} + 0.0002478\ z^{10} - 0.01882\ z^{9} - 0.0222\ z^{8} + 0.002993\ z^{7} + 0.01439\ z^{6} + 0.005272\ z^{5} - 0.0006605\ z^{4} + 0.0004409\ z^{3} + 0.0004424\ z^{2} - 0.0001467\ z$

7.3.4.3 Block διάγραμμα αποκοπής ζώνης φίλτρου (hanning).



7.3.4.4 Υλοποίηση του αλγόριθου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 34  //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[] ={
0,-1.467264e-004,4.424470e-004,4.408517e-004,
-6.605001e-004,5.271946e-003,1.438634e-002,2.992930e-003,
-2.220240e-002,-1.882488e-002,2.478253e-004,-2.712599e-002,
-5.769090e-002,3.460780e-002,1.889041e-001,1.485899e-001,
-1.235153e-001,7.085650e-001,-1.235153e-001,1.485899e-001,
1.889041e-001,3.460780e-002,-5.769090e-002,-2.712599e-002,
2.478253e-004,-1.882488e-002,-2.220240e-002,2.992930e-003,
1.438634e-002,5.271946e-003,-6.605001e-004,4.408517e-004,
4.424470e-004,-1.467264e-004,0};
static short in_buffer[N];  /*Πίνκας που έχει καθολική εμβέλια μέσα στο πρόγραμμα
                              και έχει οριστεί ώς static ετσι ώστε να διατηρεί τιμές που
                              αποθηκεύονται σε αυτόν σε προηγούμενες επαναλήψεις
                              του προγράμματος*/
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη
βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του
codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
```

```
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control
*/\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
```

Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.

```
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
```

/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */

```
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
```

{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή l_input */

```
while (!DSK6713_AIC23_read(hCodec, &l_input));
```

/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input */

```
while (!DSK6713_AIC23_read(hCodec, &r_input));
```

```
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος.*/
Ο αλγόριθμος αυτός υλοποιεί το άθροισμα
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* Πάντα κάθε νέο δείγμα εισόδου, πρίν τη διαδικασία φιλτραρισματος
                     τοποθετείται στην πρώτη θέση του πίακα in_buffer (in_buffer[0]) */
for(i=(N-1);i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* Μετακίνηση όλων των δειγμάτων κατα μία θέση δεξιά */
for(i=0;i<(N);i++)
output = output + h[i] * in_buffer[i];/*Υλοποιηση του παραπάνω αθροισματος */
return(output<<1);
}
```

7.4.1 Βαθυπερατό φίλτρο με τη μέθοδο (window κεφ.2.4)με παράθυρο blackman.

7.4.1.1 Χαρακτηριστικά φίλτρου blackman.

Βαθυπερατό FIR φίλτρο με τη μέθοδο παραθύρων τυπου (blackman)

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >74dB

Τάξη του φίλτρου N=34;

Πλάτος πλευρικου λοβού -54 dB;

Εύρος ζώνης μετάβασης Δω= 5.5/N;

Εξασθένιση στη ζώνη αποκοπής -74dB;

7.4.1.2 Υπολογισμός συνάρτησης μεταφοράς FIR φίλτρου.

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς βαθυπερατού φίλτρου (blackman):

```
n=34;
fp=5000;
fs=48000;
fn=2*fp/fs
window=blackman(n+1);
b=fir1(n,fn,window);
Hz=tf(b,1,z)
```

Τύπος συνάρτησης μεταφοράς

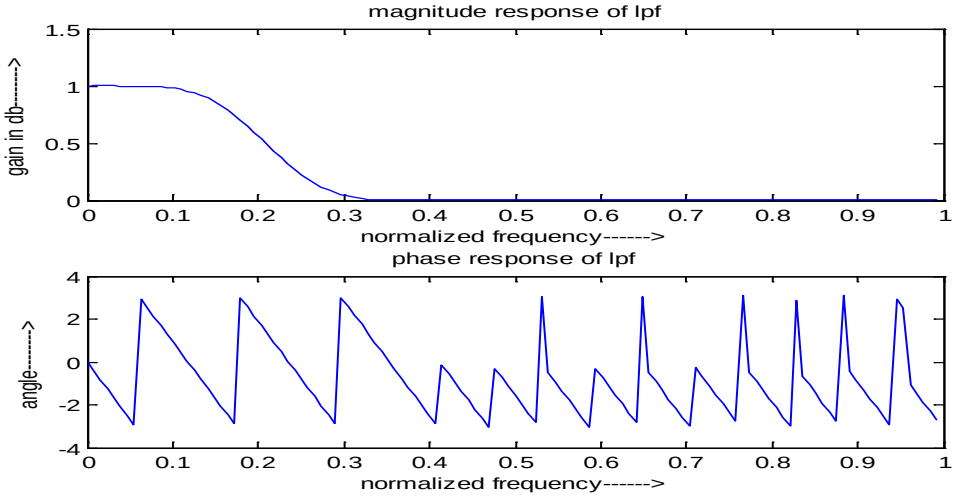
$$H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(k)z^{-k}$$

› $$H(z)=\frac{y(z)}{x(z)}=a0+a1z^{-1}+a2z^{-2}+a3z^{-3}.....+aNz^{-N}$$

Transfer function:

2.576e-019 z^{34} - 5.36e-005 z^{33} - 0.0001046 z^{32} + 0.0001798 z^{31} + 0.001124 z^{30} + 0.002567 z^{29}+ 0.003419 z^{28} + 0.001772 z^{27} - 0.003996 z^{26} - 0.01335 z^{25} - 0.02204 z^{24} - 0.02231 z^{23}- 0.005812 z^{22} + 0.03171 z^{21} + 0.08633 z^{20} + 0.1453 z^{19} + 0.1911 z^{18} + 0.2083 z^{17}+ 0.1911 z^{16} + 0.1453 z^{15} + 0.08633 z^{14} + 0.03171 z^{13} - 0.005812 z^{12} - 0.02231 z^{11}- 0.02204 z^{10} - 0.01335 z^{9} - 0.003996 z^{8} + 0.001772 z^{7} + 0.003419 z^{6} + 0.002567 z^{5}+ 0.001124 z^{4} + 0.0001798 z^{3} - 0.0001046 z^{2} - 5.36e-005 z + 2.576e-019

7.4.1.3 Block διάγραμμα βαθυπερατού φίλτρου (blackman).



7.4.1.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 34  //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[] ={
2.576151e-019,-5.360113e-005,-1.046287e-004,1.797703e-004,
1.124244e-003,2.567314e-003,3.418681e-003,1.772439e-003,
-3.995637e-003,-1.335179e-002,-2.203665e-002,-2.230656e-002,
-5.811654e-003,3.170571e-002,8.633498e-002,1.453252e-001,
1.910700e-001,2.083243e-001,1.910700e-001,1.453252e-001,
8.633498e-002,3.170571e-002,-5.811654e-003,-2.230656e-002,
-2.203665e-002,-1.335179e-002,-3.995637e-003,1.772439e-003,
3.418681e-003,2.567314e-003,1.124244e-003,1.797703e-004,
-1.046287e-004,-5.360113e-005,2.576151e-019};
static short in_buffer[N];  /*Πίνκας που έχει καθολική εμβέλια μέσα στο πρόγραμμα
                              και έχει οριστεί ώς static ετσι ώστε να διατηρεί τιμές που
                              αποθηκεύονται σε αυτόν σε προηγούμενες επαναλήψεις
                             του προγράμματος*/
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control
*/\
```

```
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
```

```
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος.*/
Ο αλγόριθμος αυτός υλοποιεί το άθροισμα
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* Πάντα κάθε νέο δείγμα εισόδου, πρίν τη διαδικασία φιλτραρισματος
                     τοποθετείται στην πρώτη θέση του πίακα in_buffer (in_buffer[0]) */
for(i=(N-1);i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* Μετακίνηση όλων των δειγμάτων κατα μία θέση δεξιά */
for(i=0;i<(N);i++)
output = output + h[i] * in_buffer[i];/*Υλοποιηση του παραπάνω αθροισματος */
return(output<<1);
}
```

7.4.2 Υψηπερατό φίλτρο με τη μέθοδο (window κεφ.2.4)με παράθυρο blackman.

7.4.2.1 Χαρακτηριστικά φίλτρου blackman.

Υψηπερατό FIR φίλτρο με τη μέθοδο παραθύρων τυπου (hanning)
Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=7KHz;
Συχνότητα δειγματοληψίας Fs=48000KHz;
Εξασθένιση στη ζώνης αποκοπής >74dB
Τάξη του φίλτρου N=34;
Πλάτος πλευρικου λοβού -54 dB;
Εύρος ζώνης μετάβασης Δω= 5.5/N;
Εξασθένιση στη ζώνη αποκοπής -74dB;

7.3.2.2 Υπολογισμός συνάρτησης μεταφοράς FIR φίλτρου.

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς υψηπερατού φίλτρου (blackman):

```
n=34;
fp=7000;
fs=48000;
fn=2*fp/fs;
window=blackman(n+1);
b=fir1(n,fn,'high',window);
```

Hz=tf(b,1,z)

Τύπος συνάρτησης μεταφοράς

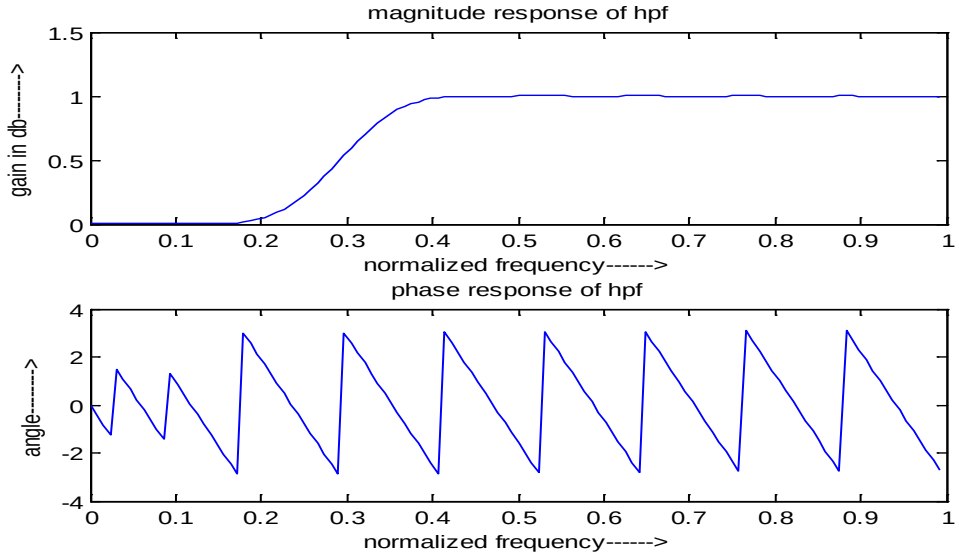
› $H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(k)z^{-k}$ (3.1)

› $H(z)=\frac{y(z)}{x(z)}=a0+a1z^{-1}+a2z^{-2}+a3z^{-3}+aNz^{-N}$

Transfer function:

3.392e-020 z^34 - 5.36e-005 z^33 - 0.0002526 z^32 - 0.0001798 z^31 + 0.0008627 z^30 + 0.002567 z^29+ 0.002623 z^28 - 0.001772 z^27 - 0.009647 z^26 - 0.01335 z^25 - 0.002901 z^24 + 0.02231 z^23+ 0.04415 z^22 + 0.03171 z^21 - 0.03576 z^20 - 0.1453 z^19 - 0.249 z^18 + 0.7083 z^17- 0.249 z^16 - 0.1453 z^15 - 0.03576 z^14 + 0.03171 z^13 + 0.04415 z^12 + 0.02231 z^11- 0.002901 z^10 - 0.01335 z^9 - 0.009647 z^8 - 0.001772 z^7 + 0.002623 z^6 + 0.002567 z^5+ 0.0008627 z^4 - 0.0001798 z^3 - 0.0002526 z^2 - 5.36e-005 z + 3.392e-020

7.4.2.3 Block διάγραμμα υψηπερατού φίλτρου (blackman).



7.3.2.4 Υλοποίηση του αλγόριθμου του φίλτρου

#include "tonecfg.h"

#include "dsk6713.h"

#include "dsk6713_aic23.h"

#define N 34 //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1

/*Ο πίνακας με τις σταθερές του φίλτρου */

float filter_Coeff[] ={

3.391656e-020,-5.360253e-005,-2.526026e-004,-1.797750e-004,

8.626858e-004,2.567381e-003,2.623315e-003,-1.772485e-003,

```
-9.646574e-003,-1.335214e-002,-2.901257e-003,2.230715e-002,
4.414505e-002,3.170655e-002,-3.576206e-002,-1.453290e-001,
-2.490138e-001,7.083213e-001,-2.490138e-001,-1.453290e-001,
-3.576206e-002,3.170655e-002,4.414505e-002,2.230715e-002,
-2.901257e-003,-1.335214e-002,-9.646574e-003,-1.772485e-003,
2.623315e-003,2.567381e-003,8.626858e-004,-1.797750e-004,
-2.526026e-004,-5.360253e-005,3.391656e-020};
static short in_buffer[N];  /*Πίνκας που έχει καθολική εμβέλια μέσα στο πρόγραμμα
                              και έχει οριστεί ώς static ετσι ώστε να διατηρεί τιμές που
                              αποθηκεύονται σε αυτόν σε προηγούμενες επαναλήψεις
                              του προγράμματος.
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.
*/
void main()
{
```

```
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
```

/*Αλγόριθμος φιλτραρίσματος.*/

Ο αλγόριθμος αυτός υλοποιεί το άθροισμα

```
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
```

```
in_buffer[0] = x; /* Πάντα κάθε νέο δείγμα εισόδου, πρίν τη διαδικασία φιλτραρισματος
                     τοποθετείται στην πρώτη θέση του πίακα in_buffer (in_buffer[0]) */
for(i=(N-1);i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* Μετακίνηση όλων των δειγμάτων κατα μία θέση δεξιά */
for(i=0;i<(N);i++)
output = output + h[i] * in_buffer[i];/*Υλοποιηση του παραπάνω αθροισματος */
return(output<<1);
}
```

7.4.3 Διέλευσής ζώνης φίλτρο με τη μέθοδο (window κεφ.2.4)με παράθυρο Blackman.

7.4.3.1 Χαρακτηριστικά φίλτρου blackman.

Διέλευσής ζώνης FIR φίλτρο με τη μέθοδο παραθύρων τυπου (hamming)

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα αποκοπής ωs στη ζώνη αποκοπής :ωs=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >74dB

Τάξη του φίλτρου N=34;

Πλάτος πλευρικου λοβού -54 dB;

Εύρος ζώνης μετάβασης Δω= 5.5/N;

Εξασθένιση στη ζώνη αποκοπής -74dB;

7.4.3.2 Υπολογισμός συνάρτησης μεταφοράς FIR φίλτρου.

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς διέλευσης ζώνης φίλτρου (hamming):

```
n=34;
fp=5000;
fq=7000;
fs=48000;
fl=2*fp/fs;
fu=2*fq/fs;
bw=[fl fu];
window=blackman(n+1);
b=fir1(n,bw,'bandpass',window);
Hz=tf(b,1,z)
```

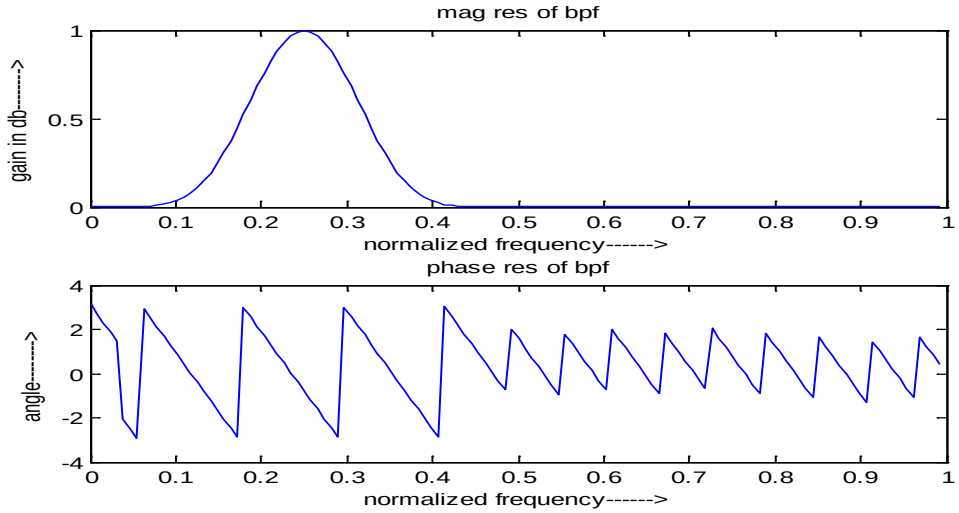
Τύπος συνάρτησης μεταφοράς

› $H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(k)z^{-k}$ (3.1)

› $H(z) = \frac{y(z)}{x(z)} = a0 + a1z^{-1} + a2z^{-2} + a3z^{-3} ..... + aNz^{-N}$

-5.319e-019 z^{34} + 0.0001956 z^{33} + 0.0006517 z^{32} + 1.112e-018 z^{31} - 0.003625 z^{30} - 0.009368 z^{29}- 0.01102 z^{28} - 3.404e-018 z^{27} + 0.02489 z^{26} + 0.04872 z^{25} + 0.0455 z^{24} - 7.528e-018 z^{23}- 0.06993 z^{22} - 0.1157 z^{21} - 0.09227 z^{20} + 0.1057 z^{18} + 0.152 z^{17} + 0.1057 z^{16}- 0.09227 z^{14} - 0.1157 z^{13} - 0.06993 z^{12} - 7.528e-018 z^{11} + 0.0455 z^{10} + 0.04872 z^{9}+ 0.02489 z^{8} - 3.404e-018 z^{7} - 0.01102 z^{6} - 0.009368 z^{5} - 0.003625 z^{4} + 1.112e-018 z^{3}+ 0.0006517 z^{2} + 0.0001956 z - 5.319e-019

7.4.3.3 Block διάγραμμα Διέλευσης ζώνης φίλτρου (blackman).



7.4.3.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 34  //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[] ={
-5.318632e-019,1.955778e-004,6.517138e-004,1.111842e-018,
-3.624877e-003,-9.367519e-003,-1.102278e-002,-3.404401e-018,
2.488813e-002,4.871752e-002,4.549611e-002,-7.527567e-018,
-6.993262e-002,-1.156866e-001,-9.226600e-002,0,
1.056997e-001,1.520252e-001,1.056997e-001,0,
-9.226600e-002,-1.156866e-001,-6.993262e-002,-7.527567e-018,
4.549611e-002,4.871752e-002,2.488813e-002,-3.404401e-018,
-1.102278e-002,-9.367519e-003,-3.624877e-003,1.111842e-018,
```

```
6.517138e-004,1.955778e-004,-5.318632e-019};
static short in_buffer[N];  /*Πίνκας που έχει καθολική εμβέλια μέσα στο πρόγραμμα
                              και έχει οριστεί ώς static ετσι ώστε να διατηρεί τιμές που
                              αποθηκεύονται σε αυτόν σε προηγούμενες επαναλήψεις
                              του προγράμματος*/
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη
βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του
codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control
*/\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου
με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του
φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
```

```
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος.*/
Ο αλγόριθμος αυτός υλοποιεί το άθροισμα
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* Πάντα κάθε νέο δείγμα εισόδου, πρίν τη διαδικασία φιλτραρισματος
                     τοποθετείται στην πρώτη θέση του πίακα in_buffer (in_buffer[0]) */
for(i=(N-1);i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* Μετακίνηση όλων των δειγμάτων κατα μία θέση δεξιά */
for(i=0;i<(N);i++)
output = output + h[i] * in_buffer[i];/*Υλοποιηση του παραπάνω αθροισματος */
return(output<<1);
```

7.4.4 Αποκοπής ζώνης φίλτρο με τη μέθοδο (window κεφ.2.4)με παράθυρο blackman.

7.4.4.1 Χαρακτηριστικά φίλτρου blackman.

Διέλευσής ζώνης FIR φίλτρο με τη μέθοδο παραθύρων τυπου (blackman)

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα αποκοπής ωs στη ζώνη αποκοπής :ωs=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >74dB

Τάξη του φίλτρου N=34;

Πλάτος πλευρικου λοβού -54 dB;

Εύρος ζώνης μετάβασης Δω= 5.5/N;

Εξασθένιση στη ζώνη αποκοπής -74dB;

7.4.4.2 Υπολογισμός συνάρτησης μεταφοράς FIR φίλτρου.

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς αποκοπής ζώνης φίλτρου (hanning):

```
n=34;
fp=5000;
fq=12000;
fs=48000;
fl=2*fp/fs;
fu=2*fq/fs;
bw=[fl fu];
window=blackman(n+1);
b=fir1(n,bw,'stop',window);
Hz=tf(b,1,z)
```

Τύπος συνάρτησης μεταφοράς

› $H(z)=\frac{y(z)}{x(z)}=\sum_{k=0}^{N}h(k)z^{-k}$ (3.1)

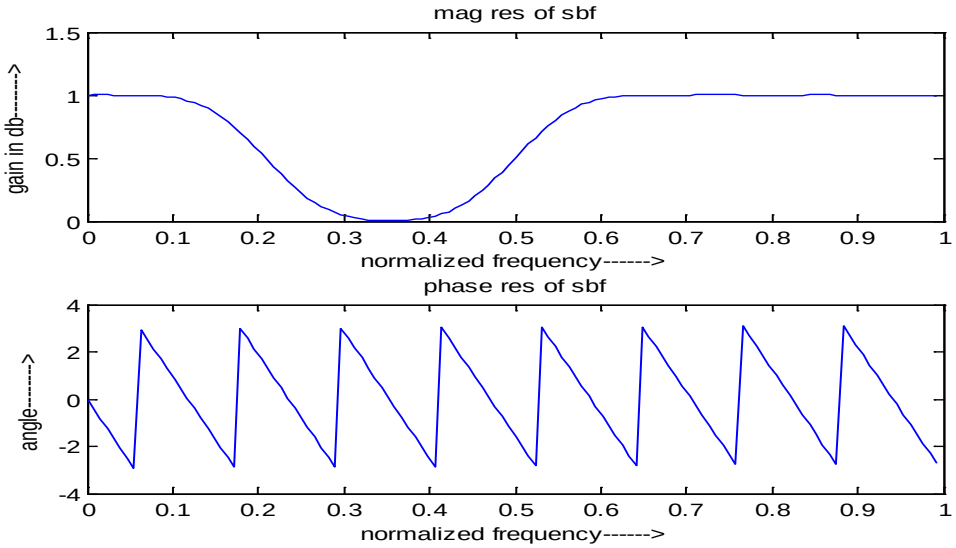
› $H(z)=\frac{y(z)}{x(z)}=a0+a1z^{-1}+a2z^{-2}+a3z^{-3}.....+aNz^{-N}$

Transfer function:

5.175e-019 z^{34} - 5.36e-005 z^{33} + 0.0001688 z^{32} + 0.0001798 z^{31} - 0.0002928 z^{30} + 0.002567 z^{29}+ 0.007728 z^{28} + 0.001772 z^{27} - 0.01444 z^{26} - 0.01335 z^{25} + 0.0001902 z^{24} - 0.02231 z^{23}- 0.05034 z^{22} + 0.03171 z^{21} + 0.1798 z^{20} + 0.1453 z^{19} - 0.1228 z^{18} + 0.7083 z^{17}- 0.1228 z^{16} + 0.1453 z^{15} + 0.1798 z^{14} + 0.03171 z^{13} - 0.05034 z^{12} - 0.02231 z^{11}+ 0.0001902 z^{10} - 0.01335 z^{9} -

$0.01444\ z^8 + 0.001772\ z^7 + 0.007728\ z^6 + 0.002567\ z^5 - 0.0002928\ z^4 + 0.0001798\ z^3 + 0.0001688\ z^2 - 5.36e\text{-}005\ z + 5.175e\text{-}019$

7.2.4.3 Block διάγραμμα αποκοπής ζώνης φίλτρου (blackman).



7.2.4.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 34  //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[] ={
5.174561e-019,-5.360144e-005,1.687802e-004,1.797713e-004,
-2.928364e-004,2.567329e-003,7.727878e-003,1.772449e-003,
-1.443682e-002,-1.335187e-002,1.901545e-004,-2.230670e-002,
-5.033676e-002,3.170590e-002,1.797844e-001,1.453261e-001,
-1.227975e-001,7.083069e-001,-1.227975e-001,1.453261e-001,
1.797844e-001,3.170590e-002,-5.033676e-002,-2.230670e-002,
1.901545e-004,-1.335187e-002,-1.443682e-002,1.772449e-003,
7.727878e-003,2.567329e-003,-2.928364e-004,1.797713e-004,
1.687802e-004,-5.360144e-005,5.174561e-019};
static short in_buffer[N];  /*Πίνκας που έχει καθολική εμβέλια μέσα στο πρόγραμμα
                              και έχει οριστεί ώς static ετσι ώστε να διατηρεί τιμές που
                              αποθηκεύονται σε αυτόν σε προηγούμενες επαναλήψεις
                              του προγράμματος*/
```

```
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
```

```
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος.*/
Ο αλγόριθμος αυτός υλοποιεί το άθροισμα
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* Πάντα κάθε νέο δείγμα εισόδου, πρίν τη διαδικασία φιλτραρισματος
                     τοποθετείται στην πρώτη θέση του πίακα in_buffer (in_buffer[0]) */
for(i=(N-1);i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* Μετακίνηση όλων των δειγμάτων κατα μία θέση δεξιά */
for(i=0;i<(N);i++)
output = output + h[i] * in_buffer[i];/*Υλοποιηση του παραπάνω αθροισματος */
return(output<<1);
}
```

ΚΕΦΑΛΑΙΟ 8
Υλοποίηση ψηφιακών φίλτρων IIR

8.1.1Βαθυπερατό φίλτρο Butterworth με μετασχηματισμό s→z με τις συναρτήσεις του Signal Processing Toolbox .

8.1.1.1Χαρακτηριστικά φίλτρου Butterworth

Βαθυπερατό IIR φίλτρο Butterworth

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >3dB

Τάξη του φίλτρου N=2;

8.1.1.2 Υπολογισμός συνάρτησης μεταφοράς

Για τον υπολογισμό της συνάρτης μεταφοράς χρησιμοποιούμε το πιο κάτω πρόγραμμα στο MATLAB:

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς βαθυπερατού φίλτρου:

```
ws=5000;
fs=48000;
wn=2*ws/fs;
wn=2*ws/fs;
[b,a]=butter(2,wn,'low');
[h,omega]=freqz(b,a);
a=a.*2^15;
b=b.*2^15;
a=round(a);
b=round(b);
a(2)=round(a(2)./2);
b(2)=round(b(2)./2);
Hz=tf(b,a,'z')
```

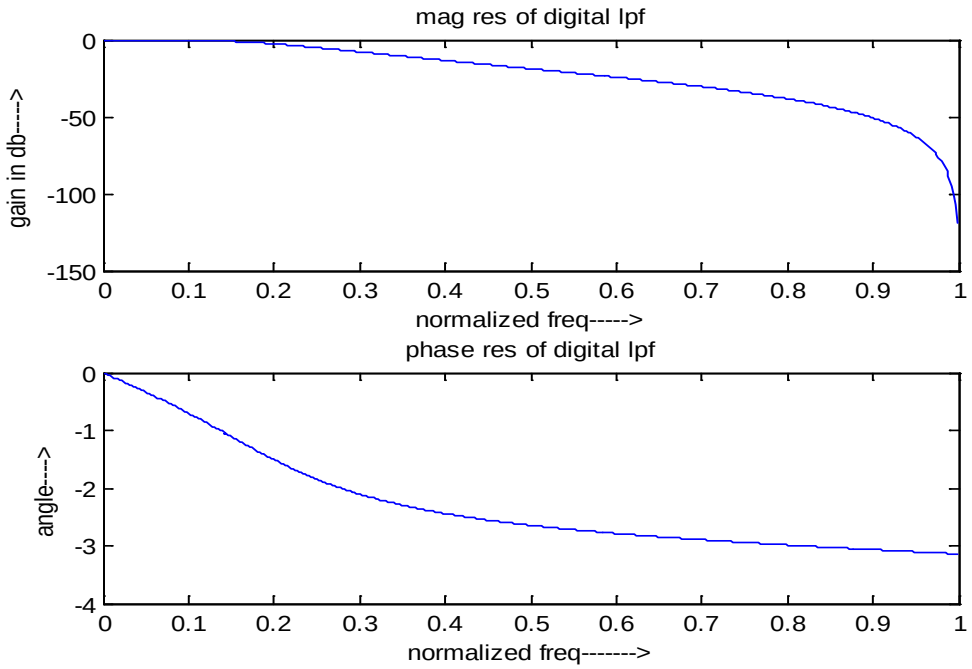
Transfer function:

$$\frac{2367\,z^2 + 2367\,z + 2367}{32768\,z^2 - 18174\,z + 13047}$$

Τύπος συνάρτησης μεταφοράς

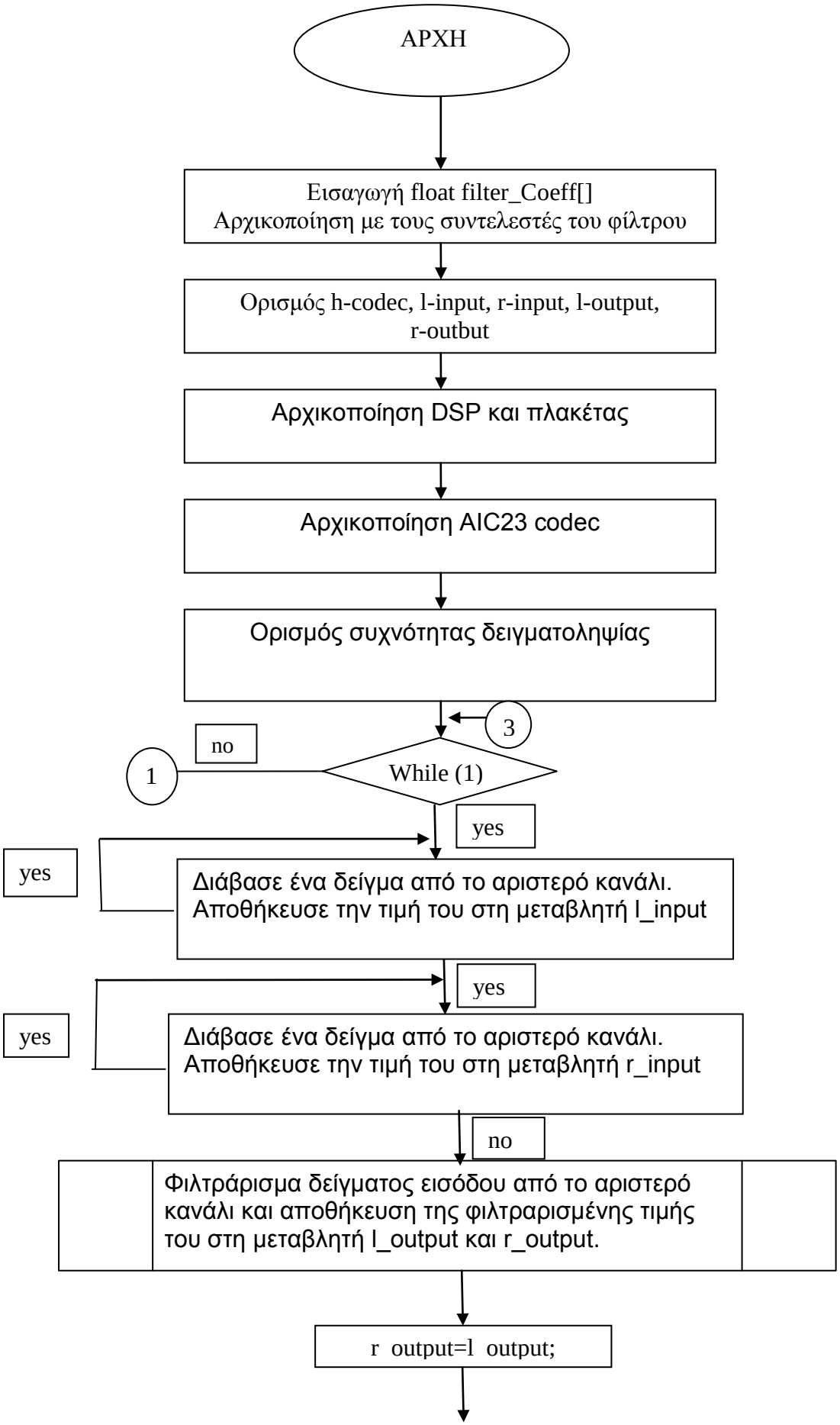
$$(2.6) H(z) = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{1 + \sum_{k=1}^{M} a(k) z^{-k}}$$

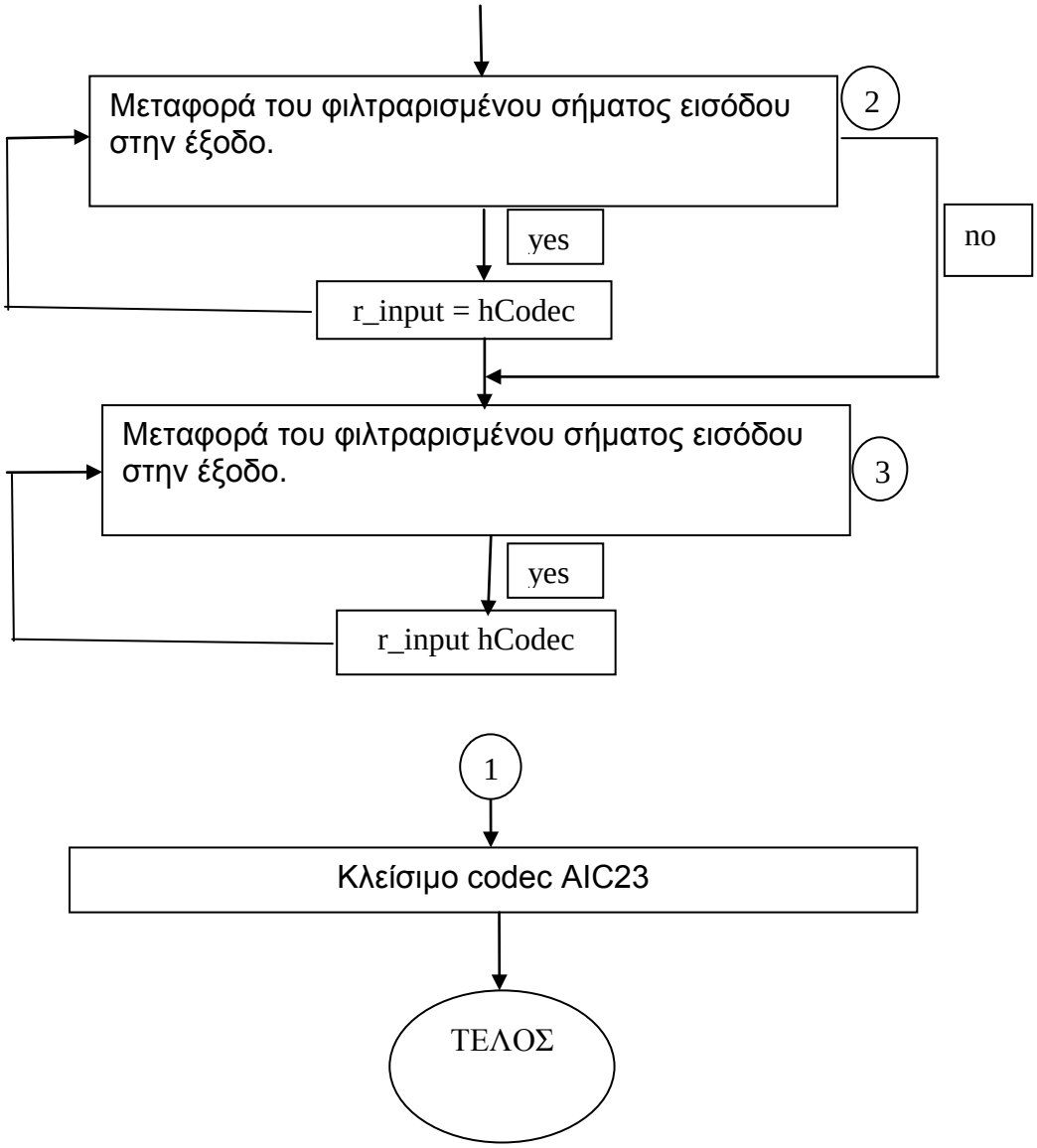
8.1.1.3 Block διάγραμμα βαθυπερατού φίλτρου Butterworth

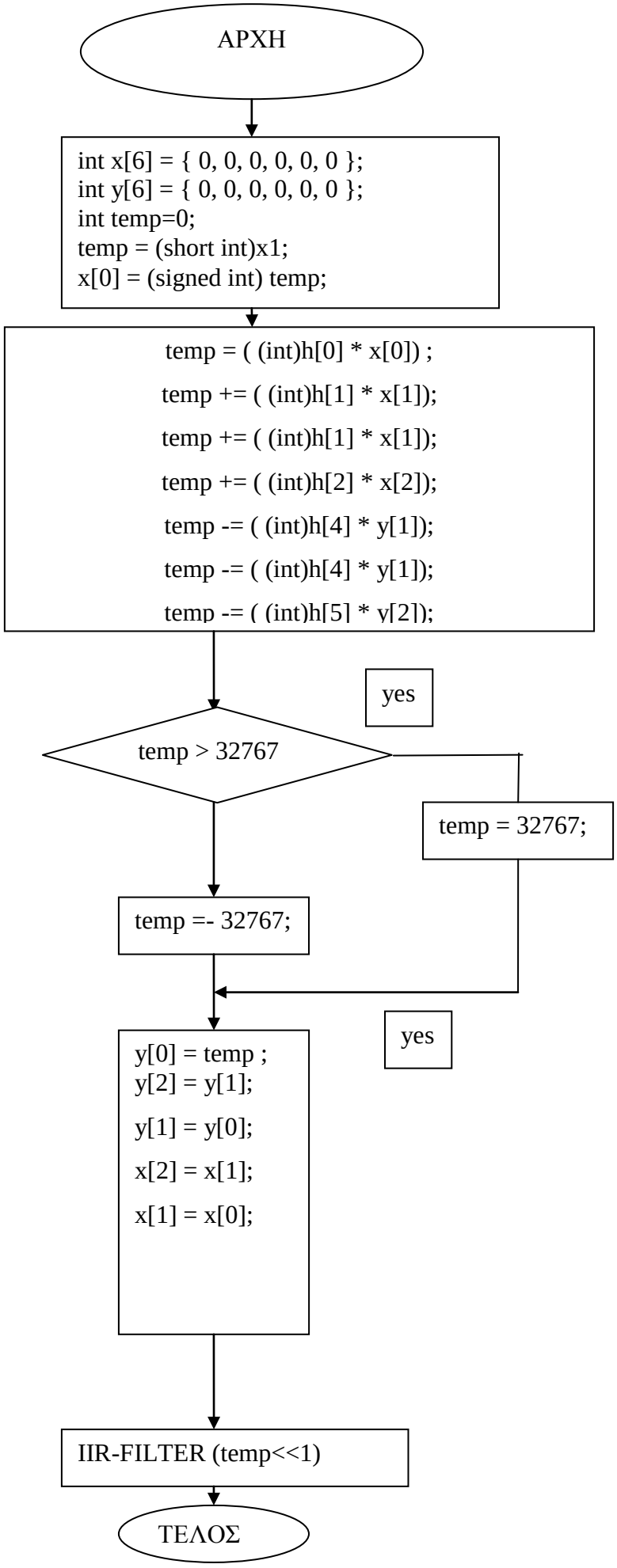


8.1.1.4 Υλοποίηση του αλγόριθμου του φίλτρου

Πιο κάτω φένεται το διάγραμμα ροής σε κώδικα C της (Texas Instruments)για IIR-FILTER. Το διάγραμμα ροής και ο κώδικας είναι ο ίδιος για όλα τα είδη IIR-FILTER,αυτό που αλλάζει κάθε φορά ανάλογα με τον τύπο του φίλτρου είναι οι συντελεστές των φίλτρων (filter coefficients)και η επιθυμητή τάξη.


ΑΡΧΗ
Εισαγωγή float filter_Coeff[]
Αρχικοποίηση με τους συντελεστές του φίλτρου
Ορισμός h-codec, l-input, r-input, l-output, r-outbut
Αρχικοποίηση DSP και πλακέτας
Αρχικοποίηση AIC23 codec
Ορισμός συχνότητας δειγματοληψίας
3
no
1
While (1)
yes
yes
Διάβασε ένα δείγμα από το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή l_input
yes
yes
Διάβασε ένα δείγμα από το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input
no
Φιλτράρισμα δείγματος εισόδου από το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητή l_output και r_output.
r_output=l_output;


Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο.
2
yes
no
r_input = hCodec
Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο.
3
yes
r_input hCodec
1
Κλείσιμο codec AIC23
ΤΕΛΟΣ


ΑΡΧΗ
int x[6] = { 0, 0, 0, 0, 0, 0 };
int y[6] = { 0, 0, 0, 0, 0, 0 };
int temp=0;
temp = (short int)x1;
x[0] = (signed int) temp;
temp = ((int)h[0] * x[0]) ;
temp += ((int)h[1] * x[1]);
temp += ((int)h[1] * x[1]);
temp += ((int)h[2] * x[2]);
temp -= ((int)h[4] * y[1]);
temp -= ((int)h[4] * y[1]);
temp -= ((int)h[5] * y[2]);
yes
temp > 32767
temp = 32767;
temp =- 32767;
yes
y[0] = temp ;
y[2] = y[1];
y[1] = y[0];
x[2] = x[1];
x[1] = x[0];
IIR-FILTER (temp<<1)
ΤΕΛΟΣ

8.1.2 Υψυπερατό φίλτρο Butterworth

8.1.2.1 Χαρακτηριστικά φίλτρου Butterworth

Υψυπερατό IIR φίλτρο Butterworth

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >3dB

Τάξη του φίλτρου N=2;

8.1.2.2 Υπολογισμός συνάρτησης μεταφοράς

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς υψυπερατού φίλτρου:

```
ws=7000;
fs=48000;
wn=2*ws/fs;
wn=2*ws/fs;
[b,a]=butter(2,wn,'high');
[h,omega]=freqz(b,a);
a=a.*2^15;
b=b.*2^15;
a=round(a);
b=round(b);
a(2)=round(a(2)./2);
b(2)=round(b(2)./2);
Hz=tf(b,a,'z')
```

Transfer function:

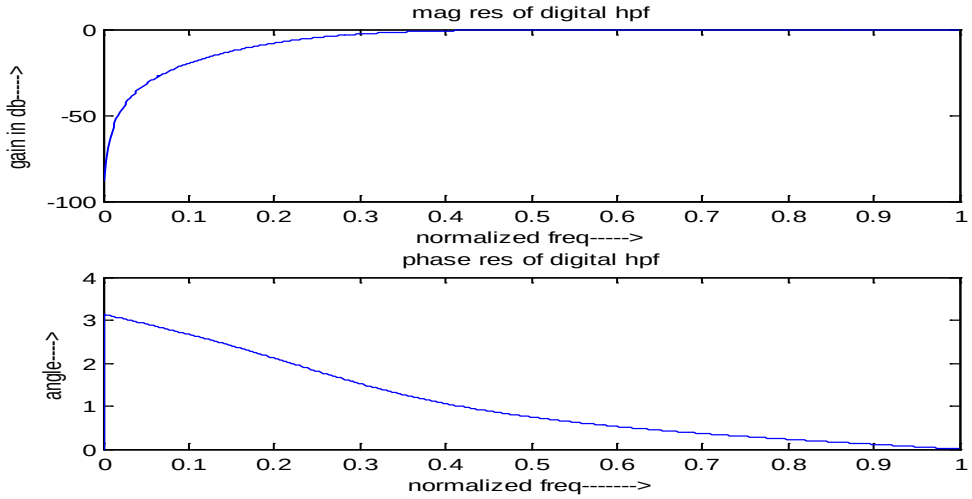
$16885\ z^2 - 16886\ s + 16885$

$32768\ z^2 - 12779\ z + 9216$

Τύπος συνάρτησης μεταφοράς

$$(2.6) H(z) = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{1 + \sum_{k=1}^{M} a(k) z^{-k}}$$

8.1.2.3 Block διάγραμμα υψυπερατού φίλτρου Butterworth



8.1.2.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 2 //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[]={
//16885,-16886,16885,32768,-12779,9216/*HPF 7000 */
} ;
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
```

```
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
```

```
}
/*Αλγόριθμος φιλτραρίσματος. Υλοποιεί φίλτρα δεύτερης τάξης*/
signed int IIR_FILTER(const signed int * h, signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Πρέπει να δειλωθεί ώς τύπου static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Πρέπει να δειλωθεί ώς τύπου static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης . Στον πίνακα y αποθηκεύονται φιλτραρισμένες τιμές απο προηγούμενα βήματα. Οι τιμές αυτές χρησιμοποιούνται στη διαδικασία φιλτραρίσματος σε επόμενα δείγματα εισόδου*/
int temp=0; /*Αρχικοποιηση του temp*/
temp = (short int)x1; /* Αντιγραφή του δείγματος εισόδου στη μεταβλητή temp */
x[0] = (signed int) temp; /* Αντιγραφή της μεταβλητής temp στην πρώτη θέση του πίνακα x. Το χ(0) εχει τώρα την τιμή του δείγματος εισόδου x[stages][0]. Στη συνέχεια ακολουθούν τα βήματα για τον υπολογισμό του φιλτραρισμένου δείγματος εξόδου, το οποίο αποθηκεύεται στην μεταβλητή temp */
temp = ( (int)h[0] * x[0]) ; /* B0 * x(n) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[2] * x[2]); /* B2 * x(n-2) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */

/* Στο σημείο αυτό του κώδικα γίνεται η αντίστροφη διαδικασία απο αυτή της κβάντισης, που είδαμε στο πρόγραμμα του Matlab */
temp >>= 15; /* Ολύσθηση κατα 15 θέσεις δεξια, αντιστοιχεί με διαίρεση με το 2^15*/
if ( temp > 32767 )
{
temp = 32767;
}
else if ( temp < -32767)
{
temp = -32767;
```

```
}
y[0] = temp ;/* ο φιλτραρισμένο δείγμα αποθηκεύεται στην πρώτη θέση του πίνακα y*/
/* Ολύσθηση όλων των στοιχείων στους πίνακες x και y κατα 1 θέση δεξιά*/
y[2] = y[1]; /* y(n-2) = y(n-1) */
y[1] = y[0]; /* y(n-1) = y(n) */
x[2] = x[1]; /* x(n-2) = x(n-1) */
x[1] = x[0]; /* x(n-1) = x(n) */
/* temp is used as input next time through */
return (temp<<1);
}
```

8.1.3 Διέλευσης ζώνης φίλτρο Butterworth

8.1.3.1 Χαρακτηριστικά φίλτρου Butterworth

Διέλευσης ζώνης IIR φίλτρο Butterworth

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα αποκοπής ωp στη ζώνη αποκοπής:ωs=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >3dB

Τάξη του φίλτρου N=1;

8.1.3.2 Υπολογισμός συνάρτησης μεταφοράς

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς Διέλευσης ζώνης φίλτρου:

```
wp=5000;
ws=7000;
fs=48000;
wn=2*ws/fs;
wm=2*wp/fs;
[b,a]=butter(1,[wm wn],'bandpass');
[h,omega]=freqz(b,a);
a=a.*2^15;
b=b.*2^15;
a=round(a);
b=round(b);
a(2)=round(a(2)./2);
b(2)=round(b(2)./2)
Hz=tf(b,a,z)
```

Transfer function:

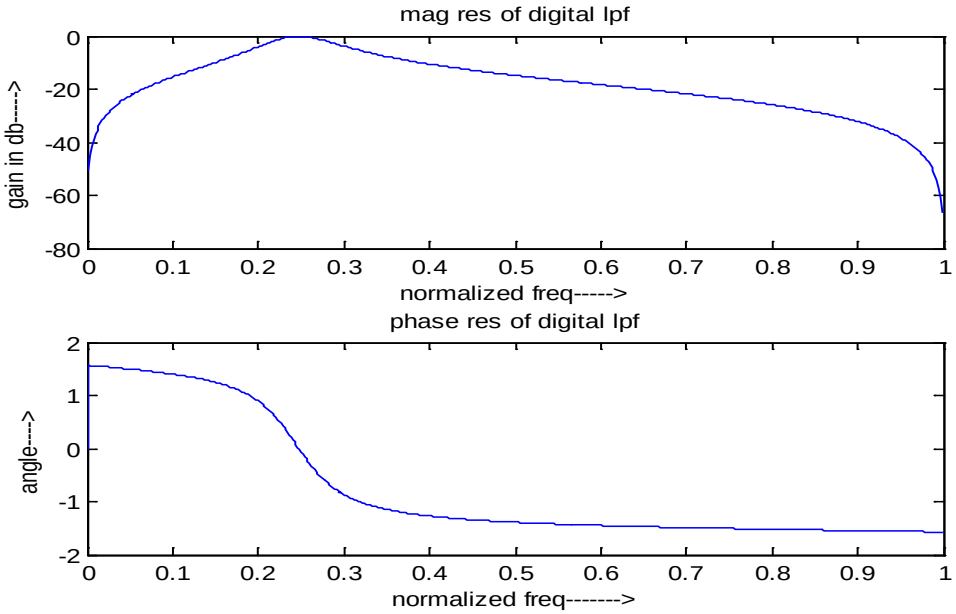
$3812\ z^2 - 3812$

$32768\ z^2 - 20652\ z + 25144$

Τύπος συνάρτησης μεταφοράς

$$(2.6) H(z) = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{1 + \sum_{k=1}^{M} a(k) z^{-k}}$$

8.1.3.3 Block διάγραμμα Διέλευσης ζώνης φίλτρου Butterworth



8.1.3.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 2 //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[]={
//3812,0,-3812,32768,-20652,25144/*BPF 5000- 7000 */
} ;
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη
βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του
codec AIC23*/
DSK6713_AIC23_Config config = {\
```

```
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control
*/\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου
με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του
φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα
δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή
l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
```

```
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της
φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος. Υλοποιεί φίλτρα δεύτερης τάξης*/
signed int IIR_FILTER(const signed int * h, signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης . Στον
πίνακα y αποθηκεύονται φιλτραρισμένες τιμές απο προηγούμενα βήματα. Οι τιμές αυτές
χρησιμοποιούνται στη διαδικασία φιλτραρίσματος σε επόμενα δείγματα εισόδου*/
int temp=0; /*Αρχικοποιηση του temp*/
temp = (short int)x1; /* Αντιγραφή του δείγματος εισόδου στη μεταβλητή temp */
x[0] = (signed int) temp; /* Αντιγραφή της μεταβλητής temp στην πρώτη θέση του πίνακα x.
Το χ(0) εχει τλωρα την τιμή του δείγματος εισόδου x[stages][0]. Στη συνέχεια ακολουθούν τα
βήματα για τον υπολογισμό του φιλτραρισμένου δείγματος εξόδου, το οποίο αποθηκεύεται
στην μεταβλητή temp */
temp = ( (int)h[0] * x[0]) ; /* B0 * x(n) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[2] * x[2]); /* B2 * x(n-2) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */
```

```
/* Στο σημείο αυτό του κώδικα γίνεται η αντίστροφη διαδικασία απο αυτή της κβάντισης, που
είδαμε στο πρόγραμμα του Matlab */
temp >>= 15; /* Ολύσθηση κατα 15 θέσεις δεξια, αντιστοιχεί με διαίρεση με το 2^15*/
if ( temp > 32767 )
{
temp = 32767;
}
else if ( temp < -32767)
{
temp = -32767;
}
y[0] = temp ;/* ο φιλτραρισμένο δείγμα αποθηκεύεται στην πρώτη θέση του πίνακα y*/
/* Ολύσθηση όλων των στοιχείων στους πίνακες x και y κατα 1 θέση δεξιά*/
y[2] = y[1]; /* y(n-2) = y(n-1) */
y[1] = y[0]; /* y(n-1) = y(n) */
x[2] = x[1]; /* x(n-2) = x(n-1) */
x[1] = x[0]; /* x(n-1) = x(n) */
/* temp is used as input next time through */
return (temp<<1);
}
```

8.1.4 Αποκοπής ζώνης φίλτρο Butterworth

8.1.4.1 Χαρακτηριστικά φίλτρου Butterworth

Αποκοπής ζώνης IIR φίλτρο Butterworth

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Εξασθένιση στη ζώνης αποκοπής >3dB

Τάξη του φίλτρου N=2;

8.1.4.2 Υπολογισμός συνάρτησης μεταφοράς

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς Αποκοπής ζώνης φίλτρου:

```
wp=5000;
ws=12000;
fs=48000;
wn=2*ws/fs;
```

```
wm=2*wp/fs;
[b,a]=butter(1,[wm wn],'stop');
[h,omega]=freqz(b,a);
a=a.*2^15;
b=b.*2^15;
a=round(a);
b=round(b);
a(2)=round(a(2)./2);
b(2)=round(b(2)./2);
Hz=tf(b,a,z)
```

Transfer function:

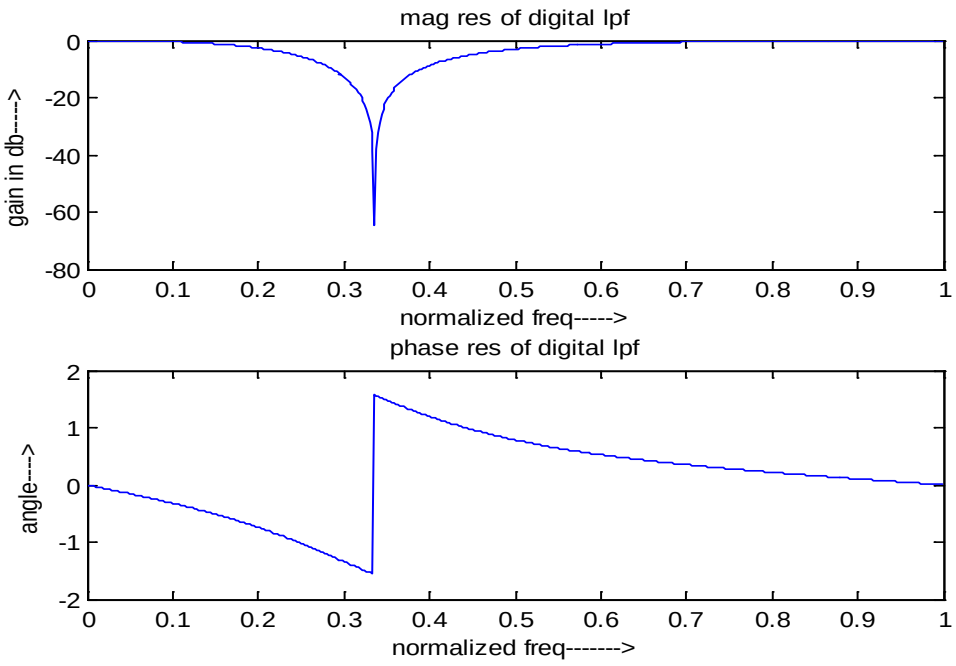
$21946\ z^2 - 10823\ z + 21946$

$32768\ z^2 - 10823\ z + 11123$

Τύπος συνάρτησης μεταφοράς

$$(2.6) H(z) = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{1 + \sum_{k=1}^{M} a(k) z^{-k}}$$

8.1.1.3 Block διάγραμμα Αποκοπής ζώνης φίλτρου Butterworth



8.1.4.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 2 //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[]={
//21946,-10823,21946,32768,-10823,11123 SPF 5000- 12000 */
} ;
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
```

```
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα
δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή
l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της
φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος. Υλοποιεί φίλτρα δεύτερης τάξης*/
signed int IIR_FILTER(const signed int * h, signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης . Στον
πίνακα y αποθηκεύονται φιλτραρισμένες τιμές απο προηγούμενα βήματα. Οι τιμές αυτές
χρησιμοποιούνται στη διαδικασία φιλτραρίσματος σε επόμενα δείγματα εισόδου*/
```

```
int temp=0; /*Αρχικοποιηση του temp*/
temp = (short int)x1; /* Αντιγραφή του δείγματος εισόδου στη μεταβλητή temp */
x[0] = (signed int) temp; /* Αντιγραφή της μεταβλητής temp στην πρώτη θέση του πίνακα x.
Το χ(0) εχει τλωρα την τιμή του δείγματος εισόδου x[stages][0]. Στη συνέχεια ακολουθούν τα
βήματα για τον υπολογισμό του φιλτραρισμένου δείγματος εξόδου, το οποίο αποθηκεύεται
στην μεταβλητή temp */
temp = ( (int)h[0] * x[0]) ; /* B0 * x(n) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[2] * x[2]); /* B2 * x(n-2) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */

/* Στο σημείο αυτό του κώδικα γίνεται η αντίστροφη διαδικασία απο αυτή της κβάντισης, που
είδαμε στο πρόγραμμα του Matlab */
temp >>= 15; /* Ολύσθηση κατα 15 θέσεις δεξια, αντιστοιχεί με διαίρεση με το 2^15*/
if ( temp > 32767 )
{
temp = 32767;
}
else if ( temp < -32767)
{
temp = -32767;
}
y[0] = temp ;/* ο φιλτραρισμένο δείγμα αποθηκεύεται στην πρώτη θέση του πίνακα y*/
/* Ολύσθηση όλων των στοιχείων στους πίνακες x και y κατα 1 θέση δεξιά*/
y[2] = y[1]; /* y(n-2) = y(n-1) */
y[1] = y[0]; /* y(n-1) = y(n) */
x[2] = x[1]; /* x(n-2) = x(n-1) */
x[1] = x[0]; /* x(n-1) = x(n) */
/* temp is used as input next time through */
return (temp<<1);
}
```

8.2.1 Βαθυπερατό φίλτρο ChebyshevI

8.2.1.1 Χαρακτηριστικά φίλτρου ChebyshevI

Βαθυπερατό IIR φίλτρο ChebyshevI

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Μέγιστη εξασθένιση στη ζώνη διάβασης rp=0.5dB

Τάξη του φίλτρου N=2;

8.2.1.2 Υπολογισμός συνάρτησης μεταφοράς

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς βαθυπερατού φίλτρου:

```
ws=5000;
fs=48000;
wn=2*ws/fs;
wn=2*ws/fs;
rp=0.5;
[b,a]=cheby1(2,rp,wn);
[h,omega]=freqz(b,a);
a=a.*2^15;
b=b.*2^15;
a=round(a);
b=round(b);
a(2)=round(a(2)./2);
b(2)=round(b(2)./2);
Hz=tf(b,a,z)
```

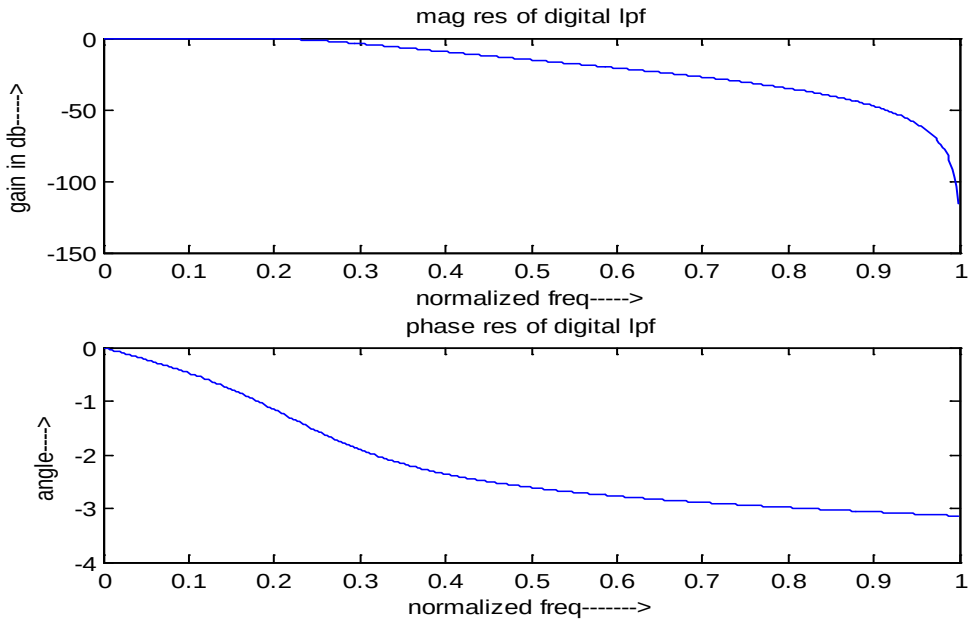
Transfer function:

$$\frac{3258\,z^2 + 3259\,z + 3258}{32768\,z^2 - 16305\,z + 13647}$$

Τύπος συνάρτησης μεταφοράς

$$(2.6)H(z) = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{1 + \sum_{k=1}^{M} a(k) z^{-k}}$$

8.2.1.3 Block διάγραμμα βαθυπερατού φίλτρου ChebyshevI



8.2.1.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 2 //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[]={
// 3258,3259,3258,32768,-16305,13647 /*LPF 5000 */
} ;
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη
βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του
codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
```

```
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control
*/\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου
με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του
φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα
δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή
l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της
φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
```

```
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος. Υλοποιεί φίλτρα δεύτερης τάξης*/
signed int IIR_FILTER(const signed int * h, signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Πρέπει να δειλωθεί ώς τύπου static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Πρέπει να δειλωθεί ώς τύπου static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης . Στον πίνακα y αποθηκεύονται φιλτραρισμένες τιμές απο προηγούμενα βήματα. Οι τιμές αυτές χρησιμοποιούνται στη διαδικασία φιλτραρίσματος σε επόμενα δείγματα εισόδου*/
int temp=0; /*Αρχικοποιηση του temp*/
temp = (short int)x1; /* Αντιγραφή του δείγματος εισόδου στη μεταβλητή temp */
x[0] = (signed int) temp; /* Αντιγραφή της μεταβλητής temp στην πρώτη θέση του πίνακα x. Το χ(0) εχει τλωρα την τιμή του δείγματος εισόδου x[stages][0]. Στη συνέχεια ακολουθούν τα βήματα για τον υπολογισμό του φιλτραρισμένου δείγματος εξόδου, το οποίο αποθηκεύεται στην μεταβλητή temp */
temp = ( (int)h[0] * x[0]) ; /* B0 * x(n) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[2] * x[2]); /* B2 * x(n-2) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */

/* Στο σημείο αυτό του κώδικα γίνεται η αντίστροφη διαδικασία απο αυτή της κβάντισης, που είδαμε στο πρόγραμμα του Matlab */
temp >>= 15; /* Ολύσθηση κατα 15 θέσεις δεξια, αντιστοιχεί με διαίρεση με το 2^15*/
if ( temp > 32767 )
{
temp = 32767;
}
```

```
else if ( temp < -32767)
{
temp = -32767;
}
y[0] = temp ;/* ο φιλτραρισμένο δείγμα αποθηκεύεται στην πρώτη θέση του πίνακα y*/
/* Ολύσθηση όλων των στοιχείων στους πίνακες x και y κατα 1 θέση δεξιά*/
y[2] = y[1]; /* y(n-2) = y(n-1) */
y[1] = y[0]; /* y(n-1) = y(n) */
x[2] = x[1]; /* x(n-2) = x(n-1) */
x[1] = x[0]; /* x(n-1) = x(n) */
/* temp is used as input next time through */
return (temp<<1);
}
```

8.2.1 Υψυπερατό φίλτρο ChebyshevI

8.2.1.1 Χαρακτηριστικά φίλτρου ChebyshevI

Υψυπερατό IIR φίλτρο ChebyshevI

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Μέγιστη εξασθένιση στη ζώνη διάβασης rp=0.5dB

Τάξη του φίλτρου N=2;

8.2.2.2 Υπολογισμός συνάρτησης μεταφοράς

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς υψυπερατού φίλτρου

```
ws=7000;
fs=48000;
wn=2*ws/fs;
rp=0.5;
[b,a]=cheby1(2,rp,wn,'high');
[h,omega]=freqz(b,a);
a=a.*2^15;
b=b.*2^15;
a=round(a);
b=round(b);
a(2)=round(a(2)./2);
b(2)=round(b(2)./2);
```

Hz=tf(b,a,z)

Transfer function:

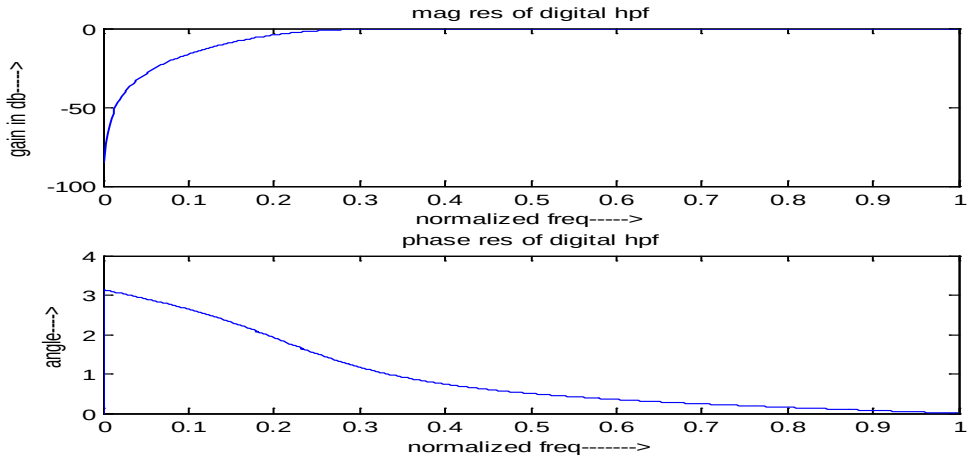
19048 z ^{2} - 19048 z + 19048

32768 z ^{2} - 16940 z + 14057

Τύπος συνάρτησης μεταφοράς

$$(2.6) H(z) = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{1 + \sum_{k=1}^{M} a(k) z^{-k}}$$

8.2.2.3 Block διάγραμμα υψυπερατού φίλτρου ChebyshevI



8.2.2.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 2 //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[]={
// 19048,-19048,19048,32768,-16940,14057 /*HPF 7000 */
} ;
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη
βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του
codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
```

```
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control
*/\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου
με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του
φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα
δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή
l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
```

```
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της
φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος. Υλοποιεί φίλτρα δεύτερης τάξης*/
signed int IIR_FILTER(const signed int * h, signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης . Στον
πίνακα y αποθηκεύονται φιλτραρισμένες τιμές απο προηγούμενα βήματα. Οι τιμές αυτές
χρησιμοποιούνται στη διαδικασία φιλτραρίσματος σε επόμενα δείγματα εισόδου*/
int temp=0; /*Αρχικοποιηση του temp*/
temp = (short int)x1; /* Αντιγραφή του δείγματος εισόδου στη μεταβλητή temp */
x[0] = (signed int) temp; /* Αντιγραφή της μεταβλητής temp στην πρώτη θέση του πίνακα x.
Το χ(0) εχει τλωρα την τιμή του δείγματος εισόδου x[stages][0]. Στη συνέχεια ακολουθούν τα
βήματα για τον υπολογισμό του φιλτραρισμένου δείγματος εξόδου, το οποίο αποθηκεύεται
στην μεταβλητή temp */
temp = ( (int)h[0] * x[0]) ; /* B0 * x(n) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[2] * x[2]); /* B2 * x(n-2) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */
```

```
/* Στο σημείο αυτό του κώδικα γίνεται η αντίστροφη διαδικασία απο αυτή της κβάντισης, που
είδαμε στο πρόγραμμα του Matlab */
temp >>= 15; /* Ολύσθηση κατα 15 θέσεις δεξια, αντιστοιχεί με διαίρεση με το 2^15*/
if ( temp > 32767 )
{
temp = 32767;
}
else if ( temp < -32767)
{
temp = -32767;
}
y[0] = temp ;/* ο φιλτραρισμένο δείγμα αποθηκεύεται στην πρώτη θέση του πίνακα y*/
/* Ολύσθηση όλων των στοιχείων στους πίνακες x και y κατα 1 θέση δεξιά*/
y[2] = y[1]; /* y(n-2) = y(n-1) */
y[1] = y[0]; /* y(n-1) = y(n) */
x[2] = x[1]; /* x(n-2) = x(n-1) */
x[1] = x[0]; /* x(n-1) = x(n) */
/* temp is used as input next time through */
return (temp<<1);
}
```

8.2.3 Διέλευσης ζώνης φίλτρο ChebyshevI

8.2.3.1 Χαρακτηριστικά φίλτρου ChebyshevI

Διέλευσης ζώνης IIR φίλτρο ChebyshevI

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα αποκοπής ωp στη ζώνη αποκοπής:ωs=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Μέγιστη εξασθένιση στη ζώνη διάβασης rp=0.5dB

Τάξη του φίλτρου N=1;

8.1.3.2 Υπολογισμός συνάρτησης μεταφοράς

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς Διέλευσης ζώνης φίλτρου:

```
ws=7000;
wp=5000;
fs=48000;
```

```
rp=0.5;
wn=2*ws/fs;
wm=2*wp/fs;
rp=0.5;
[b,a]=cheby1(1,rp,[wm wn],'bandpass');
[h,omega]=freqz(b,a);
a=a.*2^15;
b=b.*2^15;
a=round(a);
b=round(b);
a(2)=round(a(2)./2);
b(2)=round(b(2)./2);
Hz=tf(b,a,z)
```

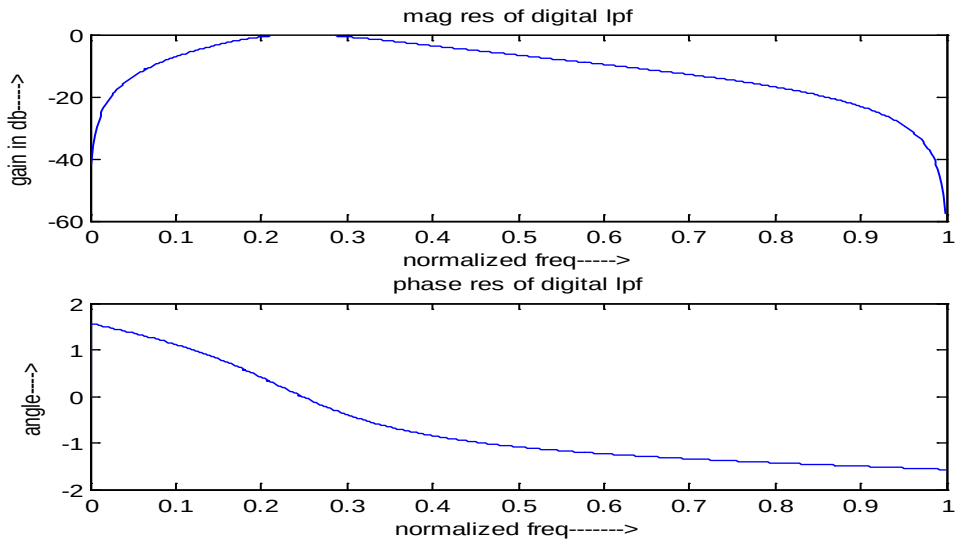
Transfer function:

$$\frac{8969\,z^{2} - 8969}{32768\,z^{2} - 16974\,z + 14829}$$

Τύπος συνάρτησης μεταφοράς

$$(2.6)H(z) = \frac{\sum_{k=0}^{N} b(k)z^{-k}}{1+\sum_{k=1}^{M} a(k)z^{-k}}$$

8.2.3.3 Block διάγραμμα Διέλευσης ζώνης φίλτρου ChebyshevI



8.2.3.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 2 //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[]={
// 8969,0,-8969,32768,-16974,14829 /*BPF 5000- 7000 */
} ;
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
```

```
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα
δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή
l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της
φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος. Υλοποιεί φίλτρα δεύτερης τάξης*/
signed int IIR_FILTER(const signed int * h, signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης . Στον
πίνακα y αποθηκεύονται φιλτραρισμένες τιμές απο προηγούμενα βήματα. Οι τιμές αυτές
χρησιμοποιούνται στη διαδικασία φιλτραρίσματος σε επόμενα δείγματα εισόδου*/
int temp=0; /*Αρχικοποιηση του temp*/
```

```
temp = (short int)x1; /* Αντιγραφή του δείγματος εισόδου στη μεταβλητή temp */
x[0] = (signed int) temp; /* Αντιγραφή της μεταβλητής temp στην πρώτη θέση του πίνακα x.
Το χ(0) εχει τλωρα την τιμή του δείγματος εισόδου x[stages][0]. Στη συνέχεια ακολουθούν τα
βήματα για τον υπολογισμό του φιλτραρισμένου δείγματος εξόδου, το οποίο αποθηκεύεται
στην μεταβλητή temp */
temp = ( (int)h[0] * x[0]) ; /* B0 * x(n) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[2] * x[2]); /* B2 * x(n-2) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */

/* Στο σημείο αυτό του κώδικα γίνεται η αντίστροφη διαδικασία απο αυτή της κβάντισης, που
είδαμε στο πρόγραμμα του Matlab */
temp >>= 15; /* Ολύσθηση κατα 15 θέσεις δεξια, αντιστοιχεί με διαίρεση με το 2^15*/
if ( temp > 32767 )
{
temp = 32767;
}
else if ( temp < -32767)
{
temp = -32767;
}
y[0] = temp ;/* ο φιλτραρισμένο δείγμα αποθηκεύεται στην πρώτη θέση του πίνακα y*/
/* Ολύσθηση όλων των στοιχείων στους πίνακες x και y κατα 1 θέση δεξιά*/
y[2] = y[1]; /* y(n-2) = y(n-1) */
y[1] = y[0]; /* y(n-1) = y(n) */
x[2] = x[1]; /* x(n-2) = x(n-1) */
x[1] = x[0]; /* x(n-1) = x(n) */
/* temp is used as input next time through */
return (temp<<1);
}
```

8.2.4 Αποκοπής ζώνης φίλτρο ChebyshevI

8.2.4.1 Χαρακτηριστικά φίλτρου ChebyshevI

Αποκοπής ζώνης IIR φίλτρο ChebyshevI

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα αποκοπής ωp στη ζώνη αποκοπής:ωp=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Μέγιστη εξασθένιση στη ζώνη διάβασης rp=0.5dB

Τάξη του φίλτρου N=1;

8.1.4.2 Υπολογισμός συνάρτησης μεταφοράς

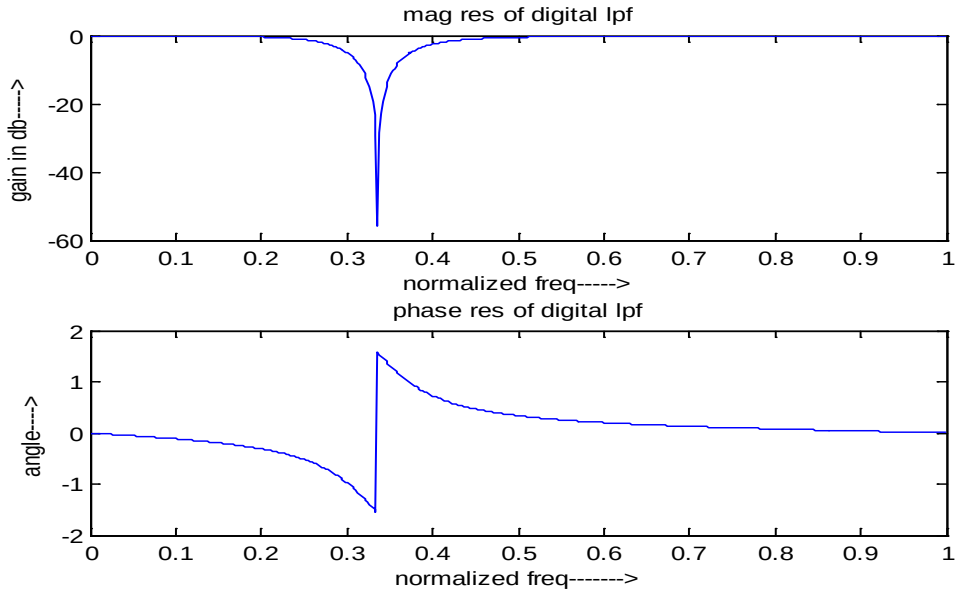
Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς Αποκοπής ζώνης φίλτρου:

```
wp=5000;
ws=12000;
fs=48000;
wn=2*ws/fs;
wm=2*wp/fs;
rp=0.5;
[b,a]=cheby1(1,rp,[wm wn],'stop');
[h,omega]=freqz(b,a);
a=a.*2^15;
b=b.*2^15;
a=round(a);
b=round(b);
a(2)=round(a(2)./2);
b(2)=round(b(2)./2);
Hz=tf(b,a,z)
Transfer function:
27953 z ^ 2 - 13785 z + 27953
---------------------------
32768 z ^ 2 - 13785 z + 23138
```

Τύπος συνάρτησης μεταφοράς

$$(2.6)H(z)=\frac{\sum_{k=0}^{N} b(k)z^{-k}}{1+\sum_{k=1}^{M} a(k)z^{-k}}$$

8.2.4.3 Block διάγραμμα Αποκοπής ζώνης φίλτρου ChebyshevI



8.2.4.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 2 //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[]={
// 27953,-13785,27953,32768,-13785,23138 SPF 5000- 12000 */
} ;
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη
βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του
codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
```

```
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control
*/\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου
με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του
φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα
δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή
l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της
φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
```

```
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος. Υλοποιεί φίλτρα δεύτερης τάξης*/
signed int IIR_FILTER(const signed int * h, signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Πρέπει να δειλωθεί ώς τύπου static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Πρέπει να δειλωθεί ώς τύπου static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης . Στον πίνακα y αποθηκεύονται φιλτραρισμένες τιμές απο προηγούμενα βήματα. Οι τιμές αυτές χρησιμοποιούνται στη διαδικασία φιλτραρίσματος σε επόμενα δείγματα εισόδου*/
int temp=0; /*Αρχικοποιηση του temp*/
temp = (short int)x1; /* Αντιγραφή του δείγματος εισόδου στη μεταβλητή temp */
x[0] = (signed int) temp; /* Αντιγραφή της μεταβλητής temp στην πρώτη θέση του πίνακα x. Το χ(0) εχει τλωρα την τιμή του δείγματος εισόδου x[stages][0]. Στη συνέχεια ακολουθούν τα βήματα για τον υπολογισμό του φιλτραρισμένου δείγματος εξόδου, το οποίο αποθηκεύεται στην μεταβλητή temp */
temp = ( (int)h[0] * x[0]) ; /* B0 * x(n) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[2] * x[2]); /* B2 * x(n-2) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */
/* Στο σημείο αυτό του κώδικα γίνεται η αντίστροφη διαδικασία απο αυτή της κβάντισης, που είδαμε στο πρόγραμμα του Matlab */
temp >>= 15; /* Ολύσθηση κατα 15 θέσεις δεξια, αντιστοιχεί με διαίρεση με το 2^15*/
if ( temp > 32767 )
{
temp = 32767;
}
else if ( temp < -32767)
```

```
{
temp = -32767;
}
y[0] = temp ;/* ο φιλτραρισμένο δείγμα αποθηκεύεται στην πρώτη θέση του πίνακα y*/
/* Ολύσθηση όλων των στοιχείων στους πίνακες x και y κατα 1 θέση δεξιά*/
y[2] = y[1]; /* y(n-2) = y(n-1) */
y[1] = y[0]; /* y(n-1) = y(n) */
x[2] = x[1]; /* x(n-2) = x(n-1) */
x[1] = x[0]; /* x(n-1) = x(n) */
/* temp is used as input next time through */
return (temp<<1);
}
```

8.3.1 Βαθυπερατό φίλτρο ChebyshevII

8.3.1.1 Χαρακτηριστικά φίλτρου ChebyshevII

Βαθυπερατό IIR φίλτρο ChebyshevII

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Μέγιστη εξασθένιση στη ζώνη αποκοπής rs=0.5dB

Τάξη του φίλτρου N=2;

8.3.1.2Υπολογισμός συνάρτησης μεταφοράς

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς βαθυπερατού φίλτρου:

```
ws=5000;
fs=48000;
wn=2*ws/fs;
rs=0.5;
[b,a]=cheby2(2,rs,wn,'low');
[h,omega]=freqz(b,a);
gain=20*log10(abs(h));
a=a.*2^15;
b=b.*2^15;
a=round(a);
b=round(b);
a(2)=round(a(2)./2);
b(2)=round(b(2)./2);
```

Hz=tf(b,a,z)

Transfer function:

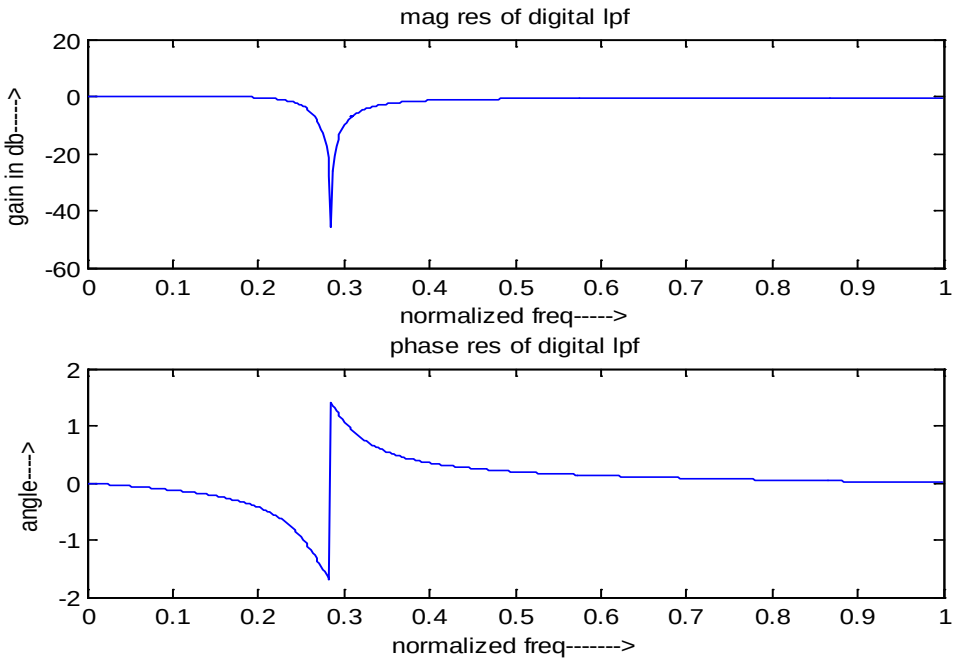
$27712\ z^{2} - 17331\ z + 27712$

$32768\ z^{2} - 18666\ z + 25324$

Τύπος συνάρτησης μεταφοράς

$$(2.6) H(z) = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{1 + \sum_{k=1}^{M} a(k) z^{-k}}$$

8.3.1.3 Block διάγραμμα βαθυπερατού φίλτρου ChebyshevII



8.3.1.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 2 //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[]={
// 27712,-17331,27712,32768,-18666,25324/*LPF 5000 */
} ;
```

```
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
```

```
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή
l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της
φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος. Υλοποιεί φίλτρα δεύτερης τάξης*/
signed int IIR_FILTER(const signed int * h, signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης . Στον
πίνακα y αποθηκεύονται φιλτραρισμένες τιμές απο προηγούμενα βήματα. Οι τιμές αυτές
χρησιμοποιούνται στη διαδικασία φιλτραρίσματος σε επόμενα δείγματα εισόδου*/
int temp=0; /*Αρχικοποιηση του temp*/
temp = (short int)x1; /* Αντιγραφή του δείγματος εισόδου στη μεταβλητή temp */
x[0] = (signed int) temp; /* Αντιγραφή της μεταβλητής temp στην πρώτη θέση του πίνακα x.
Το χ(0) εχει τλωρα την τιμή του δείγματος εισόδου x[stages][0]. Στη συνέχεια ακολουθούν τα
βήματα για τον υπολογισμό του φιλτραρισμένου δείγματος εξόδου, το οποίο αποθηκεύεται
στην μεταβλητή temp */
temp = ( (int)h[0] * x[0]) ; /* B0 * x(n) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
```

```
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[2] * x[2]); /* B2 * x(n-2) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */

/* Στο σημείο αυτό του κώδικα γίνεται η αντίστροφη διαδικασία απο αυτή της κβάντισης, που είδαμε στο πρόγραμμα του Matlab */
temp >>= 15; /* Ολύσθηση κατα 15 θέσεις δεξια, αντιστοιχεί με διαίρεση με το 2^15*/
if ( temp > 32767 )
{
temp = 32767;
}
else if ( temp < -32767)
{
temp = -32767;
}
y[0] = temp ;/* ο φιλτραρισμένο δείγμα αποθηκεύεται στην πρώτη θέση του πίνακα y*/
/* Ολύσθηση όλων των στοιχείων στους πίνακες x και y κατα 1 θέση δεξιά*/
y[2] = y[1]; /* y(n-2) = y(n-1) */
y[1] = y[0]; /* y(n-1) = y(n) */
x[2] = x[1]; /* x(n-2) = x(n-1) */
x[1] = x[0]; /* x(n-1) = x(n) */
/* temp is used as input next time through */
return (temp<<1);
}
```

8.3.3 Υψυπερατό φίλτρο ChebyshevII

8.3.3.1 Χαρακτηριστικά φίλτρου ChebyshevII

Υψυπερατό IIR φίλτρο ChebyshevII

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Μέγιστη εξασθένιση στη ζώνη αποκοπής rs=0.5dB

Τάξη του φίλτρου N=2;

8.3.3.2Υπολογισμός συνάρτησης μεταφοράς

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς υψυπερατού φίλτρου:

```
ws=7000;
fs=48000;
wn=2*ws/fs;
rs=0.5;
[b,a]=cheby2(2,rs,wn,'high');
[h,omega]=freqz(b,a);
gain=20*log10(abs(h));
a=a.*2^15;
b=b.*2^15;
a=round(a);
b=round(b);
a(2)=round(a(2)./2);
b(2)=round(b(2)./2);
Hz=tf(b,a,z)
```

Transfer function:

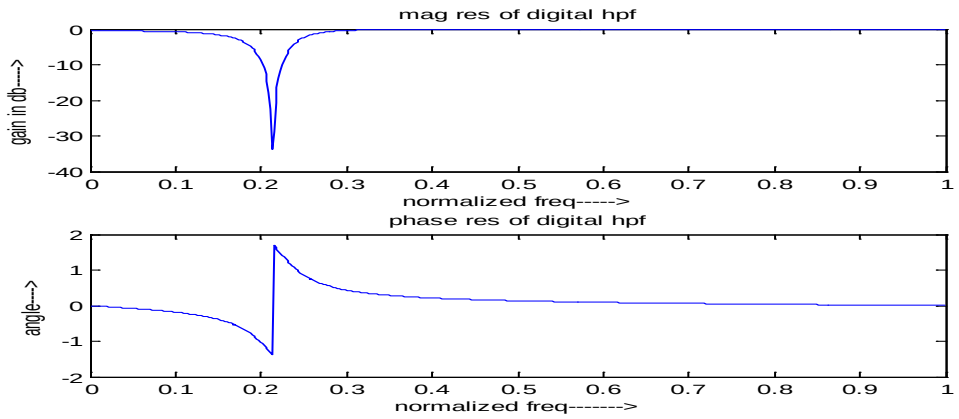
$29429\ z^{2} - 23048\ z + 29429$

$32768\ z^{2} - 22859\ z + 26469$

Τύπος συνάρτησης μεταφοράς

$$(2.6)H(z) = \frac{\sum_{k=0}^{N} b(k)z^{-k}}{1+\sum_{k=1}^{M} a(k)z^{-k}}$$

8.2.3.3 Block διάγραμμα υψυπερατού φίλτρου ChebyshevII



8.2.3.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 2 //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[]={
// 29429,-23048,29429,32768,-22859,26469/*HPF 7000 */
} ;
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
```

```
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα
δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή
l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της
φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος. Υλοποιεί φίλτρα δεύτερης τάξης*/
signed int IIR_FILTER(const signed int * h, signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης . Στον
```

πίνακα y αποθηκεύονται φιλτραρισμένες τιμές απο προηγούμενα βήματα. Οι τιμές αυτές χρησιμοποιούνται στη διαδικασία φιλτραρίσματος σε επόμενα δείγματα εισόδου*/

```
int temp=0; /*Αρχικοποιηση του temp*/
temp = (short int)x1; /* Αντιγραφή του δείγματος εισόδου στη μεταβλητή temp */
x[0] = (signed int) temp; /* Αντιγραφή της μεταβλητής temp στην πρώτη θέση του πίνακα x.
Το χ(0) εχει τλωρα την τιμή του δείγματος εισόδου x[stages][0]. Στη συνέχεια ακολουθούν τα
βήματα για τον υπολογισμό του φιλτραρισμένου δείγματος εξόδου, το οποίο αποθηκεύεται
στην μεταβλητή temp */
temp = ( (int)h[0] * x[0]) ; /* B0 * x(n) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[2] * x[2]); /* B2 * x(n-2) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */

/* Στο σημείο αυτό του κώδικα γίνεται η αντίστροφη διαδικασία απο αυτή της κβάντισης, που
είδαμε στο πρόγραμμα του Matlab */
temp >>= 15; /* Ολύσθηση κατα 15 θέσεις δεξια, αντιστοιχεί με διαίρεση με το 2^15*/
if ( temp > 32767 )
{
temp = 32767;
}
else if ( temp < -32767)
{
temp = -32767;
}
y[0] = temp ;/* ο φιλτραρισμένο δείγμα αποθηκεύεται στην πρώτη θέση του πίνακα y*/
/* Ολύσθηση όλων των στοιχείων στους πίνακες x και y κατα 1 θέση δεξιά*/
y[2] = y[1]; /* y(n-2) = y(n-1) */
y[1] = y[0]; /* y(n-1) = y(n) */
x[2] = x[1]; /* x(n-2) = x(n-1) */
x[1] = x[0]; /* x(n-1) = x(n) */
/* temp is used as input next time through */
```

return (temp<<1);

}

8.3.3 Διέλευσης ζώνης φίλτρο ChebyshevII

8.3.3.1 Χαρακτηριστικά φίλτρου ChebyshevII

Διέλευσης ζώνης IIR φίλτρο ChebyshevII

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα αποκοπής ωp στη ζώνη αποκοπής:ωs=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Μέγιστη εξασθένιση στη ζώνη αποκοπής rs=0.5dB

Τάξη του φίλτρου N=1

8.3.3.2 Υπολογισμός συνάρτησης μεταφοράς

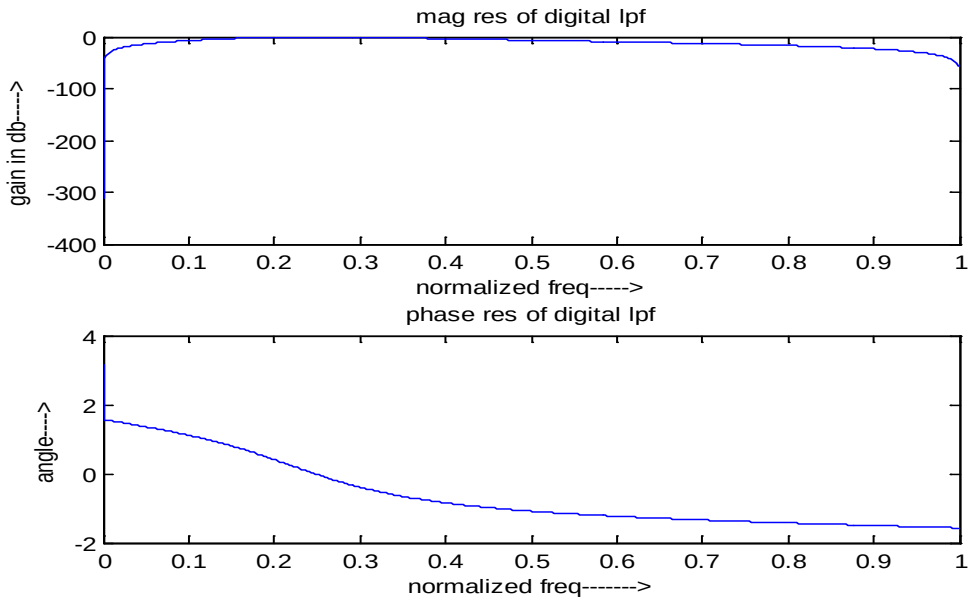
Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς Διέλευσης ζώνης φίλτρου:

```
wp=5000;
ws=7000;
fs=48000;
wn=2*ws/fs;
wm=2*wp/fs;
rs=0.5;
[b,a]=cheby2(1,rs,[wm wn],'bandpass');
[h,omega]=freqz(b,a);
a=a.*2^15;
b=b.*2^15;
a=round(a);
b=round(b);
a(2)=round(a(2)./2);
b(2)=round(b(2)./2);
Hz=tf(b,a,z)
Transfer function:
     8969 z ^ 2 - 8969
---------------------------
32768 z ^ 2 - 16974 z + 14829
```

Τύπος συνάρτησης μεταφοράς

$$(2.6) H(z) = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{1 + \sum_{k=1}^{M} a(k) z^{-k}}$$

8.2.4.3 Block διάγραμμα Διέλευσης ζώνης φίλτρου ChebyshevII



8.2.4.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 2 //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[]={
// 8969,0,-8969,32768,-16974,14829/*BPF 5000- 7000 */
} ;
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη
βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του
codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
```

```
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control
*/\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου
με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του
φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα
δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή
l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της
φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
```

```
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος. Υλοποιεί φίλτρα δεύτερης τάξης*/
signed int IIR_FILTER(const signed int * h, signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Πρέπει να δειλωθεί ώς τύπου static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Πρέπει να δειλωθεί ώς τύπου static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης . Στον πίνακα y αποθηκεύονται φιλτραρισμένες τιμές απο προηγούμενα βήματα. Οι τιμές αυτές χρησιμοποιούνται στη διαδικασία φιλτραρίσματος σε επόμενα δείγματα εισόδου*/
int temp=0; /*Αρχικοποιηση του temp*/
temp = (short int)x1; /* Αντιγραφή του δείγματος εισόδου στη μεταβλητή temp */
x[0] = (signed int) temp; /* Αντιγραφή της μεταβλητής temp στην πρώτη θέση του πίνακα x. Το χ(0) εχει τλωρα την τιμή του δείγματος εισόδου x[stages][0]. Στη συνέχεια ακολουθούν τα βήματα για τον υπολογισμό του φιλτραρισμένου δείγματος εξόδου, το οποίο αποθηκεύεται στην μεταβλητή temp */
temp = ( (int)h[0] * x[0]) ; /* B0 * x(n) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[2] * x[2]); /* B2 * x(n-2) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */

/* Στο σημείο αυτό του κώδικα γίνεται η αντίστροφη διαδικασία απο αυτή της κβάντισης, που είδαμε στο πρόγραμμα του Matlab */
temp >>= 15; /* Ολύσθηση κατα 15 θέσεις δεξια, αντιστοιχεί με διαίρεση με το 2^15*/
```

```
if ( temp > 32767 )
{
temp = 32767;
}
else if ( temp < -32767)
{
temp = -32767;
}
y[0] = temp ;/* ο φιλτραρισμένο δείγμα αποθηκεύεται στην πρώτη θέση του πίνακα y*/
/* Ολύσθηση όλων των στοιχείων στους πίνακες x και y κατα 1 θέση δεξιά*/
y[2] = y[1]; /* y(n-2) = y(n-1) */
y[1] = y[0]; /* y(n-1) = y(n) */
x[2] = x[1]; /* x(n-2) = x(n-1) */
x[1] = x[0]; /* x(n-1) = x(n) */
/* temp is used as input next time through */
return (temp<<1);
}
```

8.3.4 Αποκοπής ζώνης φίλτρο ChebyshevII

8.3.4.1 Χαρακτηριστικά φίλτρου ChebyshevII

Αποκοπής ζώνης IIR φίλτρο ChebyshevII

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα αποκοπής ωp στη ζώνη αποκοπής:ωp=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Μέγιστη εξασθένιση στη ζώνη αποκοπής rs=0.5dB

Τάξη του φίλτρου N=1;

8.3.4.2 Υπολογισμός συνάρτησης μεταφοράς

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς Αποκοπής ζώνης φίλτρου:

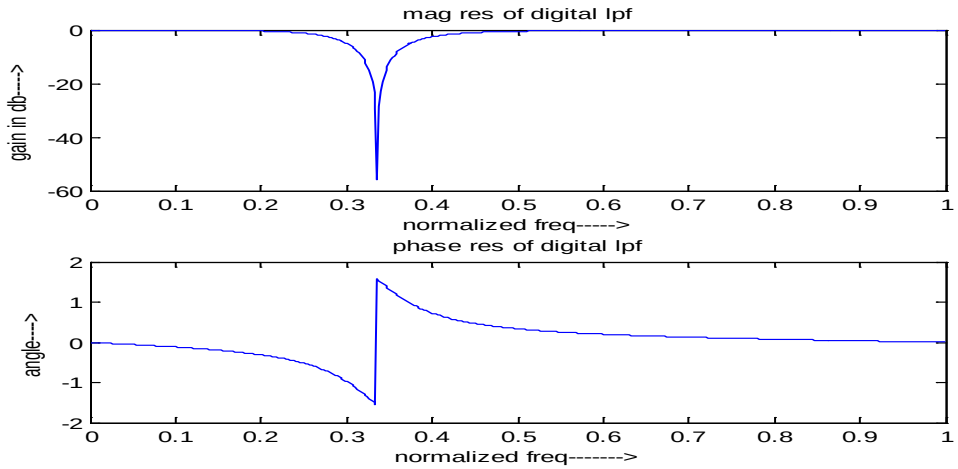
```
wp=5000;
ws=12000;
fs=48000;
wn=2*ws/fs;
wm=2*wp/fs;
rs=0.5;
[b,a]=cheby2(1,rs,[wm wn],'stop');
```

```
[h,omega]=freqz(b,a);
a=a.*2^15;
b=b.*2^15;
a=round(a);
b=round(b);
a(2)=round(a(2)./2);
b(2)=round(b(2)./2);
Hz=tf(b,a,z)
Transfer function:
27953 z ^2 - 13785 z + 27953
---------------------------
32768 z ^2 - 13785 z + 23138
```

Τύπος συνάρτησης μεταφοράς

$$(2.6) H(z) = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{1 + \sum_{k=1}^{M} a(k) z^{-k}}$$

8.3.4.2 Υπολογισμός συνάρτησης μεταφοράς



8.3.4.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 2 //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[]={
// 27953,-13785,27953,32768,-13785,23138 SPF 5000- 12000 */
```

```
} ;
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
```

```
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή
l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της
φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος. Υλοποιεί φίλτρα δεύτερης τάξης*/
signed int IIR_FILTER(const signed int * h, signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης . Στον
πίνακα y αποθηκεύονται φιλτραρισμένες τιμές απο προηγούμενα βήματα. Οι τιμές αυτές
χρησιμοποιούνται στη διαδικασία φιλτραρίσματος σε επόμενα δείγματα εισόδου*/
int temp=0; /*Αρχικοποιηση του temp*/
temp = (short int)x1; /* Αντιγραφή του δείγματος εισόδου στη μεταβλητή temp */
x[0] = (signed int) temp; /* Αντιγραφή της μεταβλητής temp στην πρώτη θέση του πίνακα x.
Το χ(0) εχει τλωρα την τιμή του δείγματος εισόδου x[stages][0]. Στη συνέχεια ακολουθούν τα
βήματα για τον υπολογισμό του φιλτραρισμένου δείγματος εξόδου, το οποίο αποθηκεύεται
στην μεταβλητή temp */
temp = ( (int)h[0] * x[0]) ; /* B0 * x(n) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
```

```
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[2] * x[2]); /* B2 * x(n-2) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */
/* Στο σημείο αυτό του κώδικα γίνεται η αντίστροφη διαδικασία απο αυτή της κβάντισης, που
είδαμε στο πρόγραμμα του Matlab */
temp >>= 15; /* Ολύσθηση κατα 15 θέσεις δεξια, αντιστοιχεί με διαίρεση με το 2^15*/
if ( temp > 32767 )
{
temp = 32767;
}
else if ( temp < -32767)
{
temp = -32767;
}
y[0] = temp ;/* ο φιλτραρισμένο δείγμα αποθηκεύεται στην πρώτη θέση του πίνακα y*/
/* Ολύσθηση όλων των στοιχείων στους πίνακες x και y κατα 1 θέση δεξιά*/
y[2] = y[1]; /* y(n-2) = y(n-1) */
y[1] = y[0]; /* y(n-1) = y(n) */
x[2] = x[1]; /* x(n-2) = x(n-1) */
x[1] = x[0]; /* x(n-1) = x(n) */
/* temp is used as input next time through */
return (temp<<1);
}
```

8.4.1 Βαθυπερατό φίλτρο Elliptic με τις συναρτήσεις

8.4.1.1 Χαρακτηριστικά φίλτρου Elliptic

Βαθυπερατό IIR φίλτρο Elliptic

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Μέγιστη εξασθένιση στη ζώνη αποκοπής rs=0.5dB

Μέγιστη εξασθένιση στη ζώνη διάβασης rp=0.5dB

Τάξη του φίλτρου N=2;

8.4.1.2 Υπολογισμός συνάρτησης μεταφοράς

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς βαθυπερατού φίλτρου:

```
ws=5000;
fs=48000;
wn=2*ws/fs;
rp=0.5;
rs=60;
[b,a] = ellip(2,rp,rs,wn);
[h,omega]=freqz(b,a);
a=a.*2^15;
b=b.*2^15;
a=round(a);
b=round(b);
a(2)=round(a(2)./2);
b(2)=round(b(2)./2);
Hz=tf(b,a,'z')
```

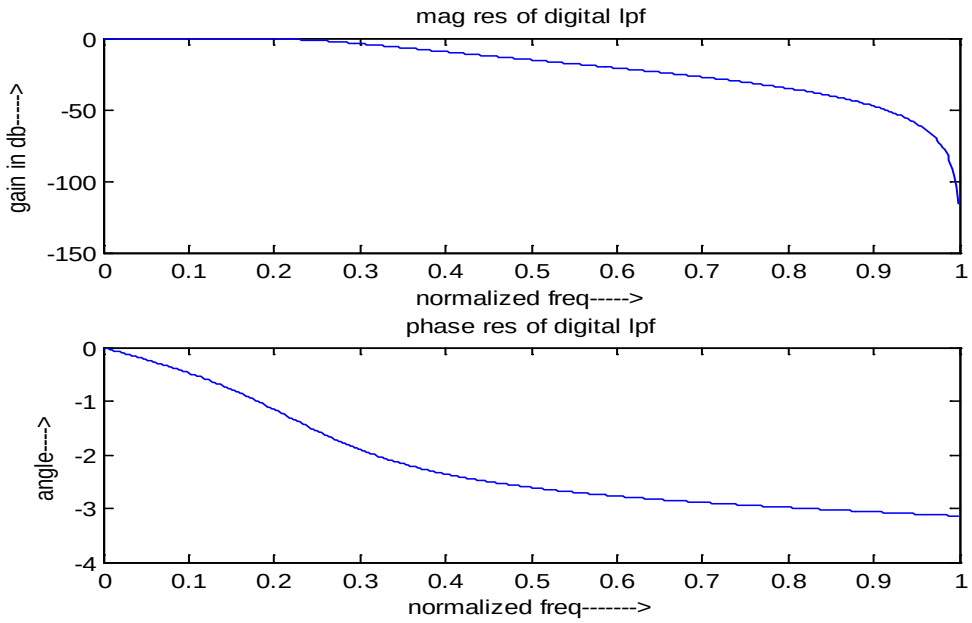
Transfer function:

$$\frac{3280\,z^2 + 3241\,z + 3280}{32768\,z^2 - 16305\,z + 1365}$$

Τύπος συνάρτησης μεταφοράς

$$(2.6)\quad H(z) = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{1 + \sum_{k=1}^{M} a(k) z^{-k}}$$

8.4.1.3 Block διάγραμμα βαθυπερατού φίλτρου Elliptic



8.4.1.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 2 //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[]={
// 3258,3259,3258,32768,-16305,13647 /*LPF 5000 */
} ;
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη
βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του
codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
```

```
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control
*/\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου
με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του
φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα
δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή
l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της
φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
```

```
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος. Υλοποιεί φίλτρα δεύτερης τάξης*/
signed int IIR_FILTER(const signed int * h, signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Πρέπει να δειλωθεί ώς τύπου static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Πρέπει να δειλωθεί ώς τύπου static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης . Στον πίνακα y αποθηκεύονται φιλτραρισμένες τιμές απο προηγούμενα βήματα. Οι τιμές αυτές χρησιμοποιούνται στη διαδικασία φιλτραρίσματος σε επόμενα δείγματα εισόδου*/
int temp=0; /*Αρχικοποιηση του temp*/
temp = (short int)x1; /* Αντιγραφή του δείγματος εισόδου στη μεταβλητή temp */
x[0] = (signed int) temp; /* Αντιγραφή της μεταβλητής temp στην πρώτη θέση του πίνακα x. Το χ(0) εχει τλωρα την τιμή του δείγματος εισόδου x[stages][0]. Στη συνέχεια ακολουθούν τα βήματα για τον υπολογισμό του φιλτραρισμένου δείγματος εξόδου, το οποίο αποθηκεύεται στην μεταβλητή temp */
temp = ( (int)h[0] * x[0]) ; /* B0 * x(n) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[2] * x[2]); /* B2 * x(n-2) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */

/* Στο σημείο αυτό του κώδικα γίνεται η αντίστροφη διαδικασία απο αυτή της κβάντισης, που είδαμε στο πρόγραμμα του Matlab */
temp >>= 15; /* Ολύσθηση κατα 15 θέσεις δεξια, αντιστοιχεί με διαίρεση με το 2^15*/
if ( temp > 32767 )
{
temp = 32767;
```

```
}
else if ( temp < -32767)
{
temp = -32767;
}
y[0] = temp ;/* ο φιλτραρισμένο δείγμα αποθηκεύεται στην πρώτη θέση του πίνακα y*/
/* Ολύσθηση όλων των στοιχείων στους πίνακες x και y κατα 1 θέση δεξιά*/
y[2] = y[1]; /* y(n-2) = y(n-1) */
y[1] = y[0]; /* y(n-1) = y(n) */
x[2] = x[1]; /* x(n-2) = x(n-1) */
x[1] = x[0]; /* x(n-1) = x(n) */
/* temp is used as input next time through */
return (temp<<1);
}
```

8.4.2 Υψυπερατό φίλτρο Elliptic

8.4.2.1 Χαρακτηριστικά φίλτρου Elliptic

Υψυπερατό IIR φίλτρο Butterworth

Συχνότητα αποκοπής ωρ στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Μέγιστη εξασθένιση στη ζώνη αποκοπής rs=0.5dB

Μέγιστη εξασθένιση στη ζώνη διάβασης rp=0.5dB

Τάξη του φίλτρου N=2;

8.4.2.2 Υπολογισμός συνάρτησης μεταφοράς

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς υψυπερατού φίλτρου:

```
ws=7000;
fs=48000;
wn=2*ws/fs;
rp=0.5;
rs=60;
[b,a] = ellip(2,rp,rs,wn,'high');
[h,omega]=freqz(b,a);
a=a.*2^15;
b=b.*2^15;
a=round(a);
```

b=round(b);

a(2)=round(a(2)./2);

b(2)=round(b(2)./2);

Hz=tf(b,a,z)

Transfer function:

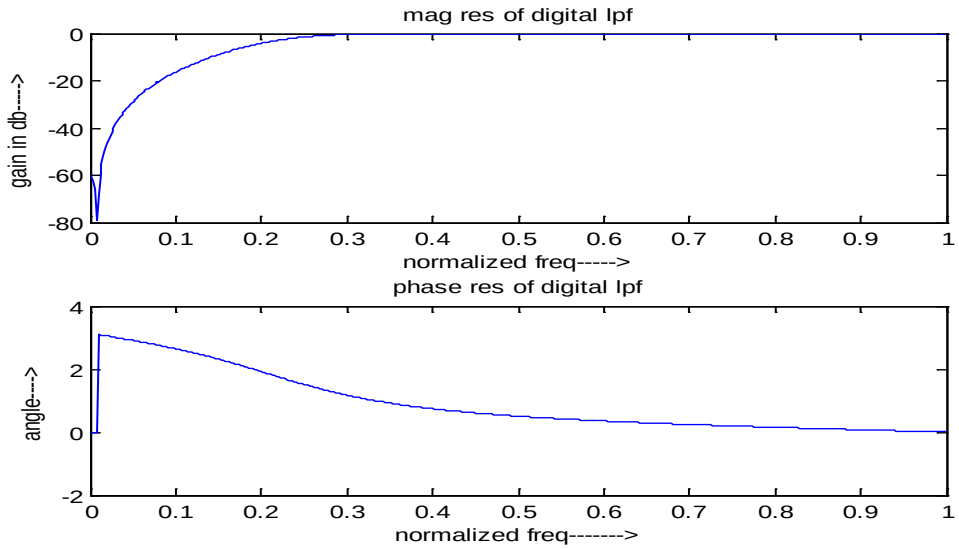
19056 z ^2 - 19050 z + 19056

32768 z ^2 - 16946 z + 14068

Τύπος συνάρτησης μεταφοράς

$$(2.6) H(z) = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{1 + \sum_{k=1}^{M} a(k) z^{-k}}$$

8.4.2.3 Block διάγραμμα υψυπερατού φίλτρου Elliptic



8.4.2.4 Υλοποίηση του αλγόριθμου του φίλτρου

#include "tonecfg.h"

#include "dsk6713.h"

#include "dsk6713_aic23.h"

#define N 2 //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1

/*Ο πίνακας με τις σταθερές του φίλτρου */

float filter_Coeff[]={

//19056,-19050,19056,32768,-16946,14068/*HPF 7000 */

```
} ;
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
```

```
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή
l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της
φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος. Υλοποιεί φίλτρα δεύτερης τάξης*/
signed int IIR_FILTER(const signed int * h, signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης . Στον
πίνακα y αποθηκεύονται φιλτραρισμένες τιμές απο προηγούμενα βήματα. Οι τιμές αυτές
χρησιμοποιούνται στη διαδικασία φιλτραρίσματος σε επόμενα δείγματα εισόδου*/
int temp=0; /*Αρχικοποιηση του temp*/
temp = (short int)x1; /* Αντιγραφή του δείγματος εισόδου στη μεταβλητή temp */
x[0] = (signed int) temp; /* Αντιγραφή της μεταβλητής temp στην πρώτη θέση του πίνακα x.
Το χ(0) εχει τλωρα την τιμή του δείγματος εισόδου x[stages][0]. Στη συνέχεια ακολουθούν τα
βήματα για τον υπολογισμό του φιλτραρισμένου δείγματος εξόδου, το οποίο αποθηκεύεται
στην μεταβλητή temp */
temp = ( (int)h[0] * x[0]) ; /* B0 * x(n) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
```

```
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[2] * x[2]); /* B2 * x(n-2) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */

/* Στο σημείο αυτό του κώδικα γίνεται η αντίστροφη διαδικασία απο αυτή της κβάντισης, που
είδαμε στο πρόγραμμα του Matlab */
temp >>= 15; /* Ολύσθηση κατα 15 θέσεις δεξια, αντιστοιχεί με διαίρεση με το 2^15*/
if ( temp > 32767 )
{
temp = 32767;
}
else if ( temp < -32767)
{
temp = -32767;
}
y[0] = temp ;/* ο φιλτραρισμένο δείγμα αποθηκεύεται στην πρώτη θέση του πίνακα y*/
/* Ολύσθηση όλων των στοιχείων στους πίνακες x και y κατα 1 θέση δεξιά*/
y[2] = y[1]; /* y(n-2) = y(n-1) */
y[1] = y[0]; /* y(n-1) = y(n) */
x[2] = x[1]; /* x(n-2) = x(n-1) */
x[1] = x[0]; /* x(n-1) = x(n) */
/* temp is used as input next time through */
return (temp<<1);
}
```

8.4.3 Διέλευσης ζώνης φίλτρο Elliptic

8.4.3.1 Χαρακτηριστικά φίλτρου Elliptic

Διέλευσης ζώνης IIR φίλτρο Elliptic

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα αποκοπής ωp στη ζώνη αποκοπής:ωs=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Μέγιστη εξασθένιση στη ζώνη αποκοπής rs=0.5dB

Μέγιστη εξασθένιση στη ζώνη διάβασης rp=0.5dB

Τάξη του φίλτρου N=1;

8.4.3.2 Υπολογισμός συνάρτησης μεταφοράς

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς Διέλευσης ζώνης φίλτρου:

```
wp=5000;
ws=7000;
fs=48000;
wn=2*ws/fs;
wm=2*wp/fs;
rp=0.5;
rs=60;
[b,a] = ellip(1,rp,rs,[wm wn],'bandpass');
[h,omega]=freqz(b,a);
a=a.*2^15;
b=b.*2^15;
a=round(a);
b=round(b);
a(2)=round(a(2)./2);
b(2)=round(b(2)./2);
Hz=tf(b,a,z)
```

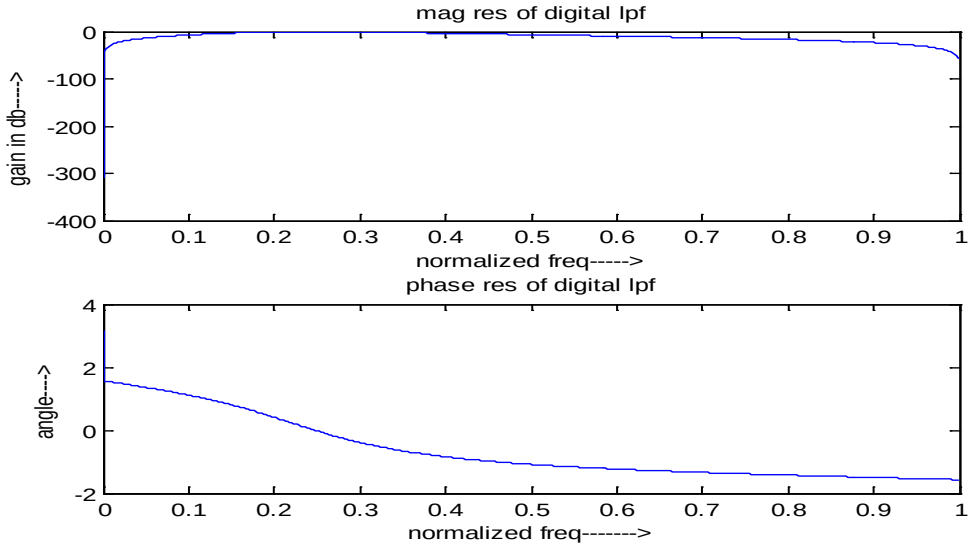
Transfer function:

$$\frac{8969\ z^2 - 8969}{32768\ z^2 - 16974\ z + 14829}$$

Τύπος συνάρτησης μεταφοράς

$$(2.6)H(z) = \frac{\sum_{k=0}^{N} b(k)z^{-k}}{1+\sum_{k=1}^{M} a(k)z^{-k}}$$

8.4.3.3 Block διάγραμμα Διέλευσης ζώνης φίλτρου Elliptic



8.4.3.4 Υλοποίηση του αλγόριθμου του φίλτρου

```
#include "tonecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#define N 1 //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1
/*Ο πίνακας με τις σταθερές του φίλτρου */
float filter_Coeff[]={
// 8969,0,-8969,32768,-16974,14829 /*BPF 5000- 7000 */
} ;
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
```

```
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control
*/\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου
με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του
φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα
δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή
l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της
φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
```

```
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος. Υλοποιεί φίλτρα δεύτερης τάξης*/
signed int IIR_FILTER(const signed int * h, signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Πρέπει να δειλωθεί ώς τύπου static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Πρέπει να δειλωθεί ώς τύπου static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης . Στον πίνακα y αποθηκεύονται φιλτραρισμένες τιμές απο προηγούμενα βήματα. Οι τιμές αυτές χρησιμοποιούνται στη διαδικασία φιλτραρίσματος σε επόμενα δείγματα εισόδου*/
int temp=0; /*Αρχικοποιηση του temp*/
temp = (short int)x1; /* Αντιγραφή του δείγματος εισόδου στη μεταβλητή temp */
x[0] = (signed int) temp; /* Αντιγραφή της μεταβλητής temp στην πρώτη θέση του πίνακα x. Το χ(0) εχει τλωρα την τιμή του δείγματος εισόδου x[stages][0]. Στη συνέχεια ακολουθούν τα βήματα για τον υπολογισμό του φιλτραρισμένου δείγματος εξόδου, το οποίο αποθηκεύεται στην μεταβλητή temp */
temp = ( (int)h[0] * x[0]) ; /* B0 * x(n) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[2] * x[2]); /* B2 * x(n-2) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */

/* Στο σημείο αυτό του κώδικα γίνεται η αντίστροφη διαδικασία απο αυτή της κβάντισης, που είδαμε στο πρόγραμμα του Matlab */
temp >>= 15; /* Ολύσθηση κατα 15 θέσεις δεξια, αντιστοιχεί με διαίρεση με το 2^15*/
if ( temp > 32767 )
{
temp = 32767;
}
```

```
else if ( temp < -32767)
{
temp = -32767;
}
y[0] = temp ;/* ο φιλτραρισμένο δείγμα αποθηκεύεται στην πρώτη θέση του πίνακα y*/
/* Ολύσθηση όλων των στοιχείων στους πίνακες x και y κατα 1 θέση δεξιά*/
y[2] = y[1]; /* y(n-2) = y(n-1) */
y[1] = y[0]; /* y(n-1) = y(n) */
x[2] = x[1]; /* x(n-2) = x(n-1) */
x[1] = x[0]; /* x(n-1) = x(n) */
/* temp is used as input next time through */
return (temp<<1);
}
```

8.4.4 Αποκοπής ζώνης φίλτρο Elliptic

8.4.4.1 Χαρακτηριστικά φίλτρου Elliptic

Αποκοπής ζώνης IIR φίλτρο Elliptic

Συχνότητα αποκοπής ωp στη ζώνη διάβασης:ωp=5KHz;

Συχνότητα αποκοπής ωp στη ζώνη αποκοπής:ωp=7KHz;

Συχνότητα δειγματοληψίας Fs=48000KHz;

Μέγιστη εξασθένιση στη ζώνη αποκοπής rs=0.5dB

Μέγιστη εξασθένιση στη ζώνη διάβασης rp=0.5dB

Τάξη του φίλτρου N=1;

8.4.4.2 Υπολογισμός συνάρτησης μεταφοράς

Πρόγραμμα υπολογισμού συνάρτησης μεταφοράς Αποκοπής ζώνης φίλτρου:

```
wp=5000;
ws=12000;
fs=48000;
wn=2*ws/fs;
wm=2*wp/fs;
rp=0.5;
rs=60;
[b,a] = ellip(1,rp,rs,[wm wn],'stop');
[h,omega]=freqz(b,a);
a=a.*2^15;
```

b=b.*2^15;

a=round(a);

b=round(b);

a(2)=round(a(2)./2);

b(2)=round(b(2)./2);

Hz=tf(b,a,z)

Transfer function:

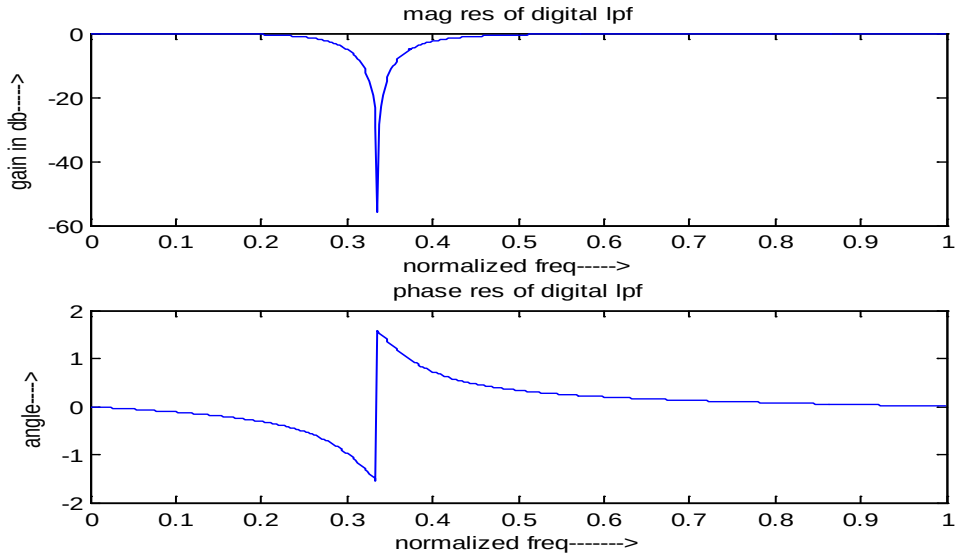
27953 z ^{2} - 13785 z + 27953

32768 z ^{2} - 13785 z + 23138

Τύπος συνάρτησης μεταφοράς

$$(2.6) H(z) = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{1 + \sum_{k=1}^{M} a(k) z^{-k}}$$

8.4.4.3 Block διάγραμμα Αποκοπής ζώνης φίλτρου Elliptic.



8.4.4.4 Υλοποίηση του αλγόριθμου του φίλτρου

#include "tonecfg.h"

#include "dsk6713.h"

#include "dsk6713_aic23.h"

#define N 2 //Πλήθος των σταθερών του φίλτρου ίσο με την τάξη του φίλτρου +1

/*Ο πίνακας με τις σταθερές του φίλτρου */

float filter_Coeff[]={

```
// 27953,-13785,27953,32768,-13785,23138 SPF 5000- 12000 */
} ;
/*Δομή τύπου DSK6713_AIC23_Config με το όνομα config. Η δομή αυτή ορίζεται μέσα στη βιβλιοθήκη  dsk6713_aic23.h και περιέχει όλες τις πληροφορίες για την αρχικοποίηση του codec AIC23*/
DSK6713_AIC23_Config config = {\
0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Leftline input channel volume */\
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume*/\
0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */\
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */\
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */\
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */\
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */\
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */\
DSK6713_AIC23_FREQ_48KHZ, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */\
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
/*
Κυρίος πρόγραμμα. Γίνεται αρχικοποίηση του DSP, του codec AIC23, λήψη σήματος εισόδου με συχνότητα δειγματοληξίας που ορίζεται μέσα στο πρόγραμμα, φιλτράρισμα και έξοδος του φιλτραρισμένου σήματος.
*/
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
/* Αρχικοπίηση DSP και πλακέτας */
DSK6713_init();
/* Αρχικοποίηση AIC23 codec */
hCodec = DSK6713_AIC23_openCodec(0, &config);
/* Ορισμός συχνότητας δειγματοληψίας. Αν παραλείψουμε αυτό το βήμα η συχνότητα δειγματοληψίας θα είναι αυτή που ορίστηκε στο config */
DSK6713_AIC23_setFreq ( hCodec , DSK6713_AIC23_FREQ_48KHZ );
```

```
while(1) //Εναρξη ατέρμονος βρόχου.
{ /* Διαβασε ένα δείγμα απο το αριστερό κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή
l_input */
while (!DSK6713_AIC23_read(hCodec, &l_input));
/* Διαβασε ένα δείγμα απο το δεξί κανάλι. Αποθήκευσε την τιμή του στη μεταβλητή r_input  */
while (!DSK6713_AIC23_read(hCodec, &r_input));
/*Φιλτράρισμα δείγματος εισόδου απο το αριστερό κανάλι και αποθήκευση της
φιλτραρισμένης τιμής του στη μεταβλητη l_output και r_output*/
l_output=(Int16)FIR_FILTER(&filter_Coeff ,l_input);
r_output=l_output;
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο*/
while (!DSK6713_AIC23_write(hCodec, l_output));
/* Μεταφορά του φιλτραρισμένου σήματος εισόδου στην έξοδο */
while (!DSK6713_AIC23_write(hCodec, r_output));
}
/* Κλείσημο codec AIC23 */
DSK6713_AIC23_closeCodec(hCodec);
}
/*Αλγόριθμος φιλτραρίσματος. Υλοποιεί φίλτρα δεύτερης τάξης*/
signed int IIR_FILTER(const signed int * h, signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Πρέπει να δειλωθεί ώς τύπου
static, ώστε να διατηρεί τις τιμές που είχε απο προηγούμενες κλήσεις της συνάρτησης . Στον
πίνακα y αποθηκεύονται φιλτραρισμένες τιμές απο προηγούμενα βήματα. Οι τιμές αυτές
χρησιμοποιούνται στη διαδικασία φιλτραρίσματος σε επόμενα δείγματα εισόδου*/
int temp=0; /*Αρχικοποιηση του temp*/
temp = (short int)x1; /* Αντιγραφή του δείγματος εισόδου στη μεταβλητή temp */
x[0] = (signed int) temp; /* Αντιγραφή της μεταβλητής temp στην πρώτη θέση του πίνακα x.
Το χ(0) εχει τλωρα την τιμή του δείγματος εισόδου x[stages][0]. Στη συνέχεια ακολουθούν τα
βήματα για τον υπολογισμό του φιλτραρισμένου δείγματος εξόδου, το οποίο αποθηκεύεται
στην μεταβλητή temp */
temp = ( (int)h[0] * x[0]) ; /* B0 * x(n) */
```

```
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[1] * x[1]); /* B1/2 * x(n-1) */
temp += ( (int)h[2] * x[2]); /* B2 * x(n-2) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[4] * y[1]); /* A1/2 * y(n-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */
/* Στο σημείο αυτό του κώδικα γίνεται η αντίστροφη διαδικασία απο αυτή της κβάντισης, που είδαμε στο πρόγραμμα του Matlab */
temp >>= 15; /* Ολύσθηση κατα 15 θέσεις δεξια, αντιστοιχεί με διαίρεση με το 2^15*/
if ( temp > 32767 )
{
temp = 32767;
}
else if ( temp < -32767)
{
temp = -32767;
}
y[0] = temp ;/* ο φιλτραρισμένο δείγμα αποθηκεύεται στην πρώτη θέση του πίνακα y*/
/* Ολύσθηση όλων των στοιχείων στους πίνακες x και y κατα 1 θέση δεξιά*/
y[2] = y[1]; /* y(n-2) = y(n-1) */
y[1] = y[0]; /* y(n-1) = y(n) */
x[2] = x[1]; /* x(n-2) = x(n-1) */
x[1] = x[0]; /* x(n-1) = x(n) */
/* temp is used as input next time through */
return (temp<<1);
}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Ψηφιακή επεξεργασία σήματος
   Monson H.Hayes
2. Εισαγωγή στην Ψηφιακή επεξεργασία σήματος
   Α.Παπαστεργίου Π.Τζέκης Α.Χατζικάιδας θεσσαλλονίκη 2008
3. Ψηφιακή επεξεργασία σημάτων και εικόνων
   Αθανάσιος Σκόρδας ,Βασίλειος Αναστασόπουλλος
4. Συστήματα Επεξεργασίας Σημάτων με DSPs
   Βαγγέλης Ζυγούρης Εργαστήριο Ηλεκτρονικής Πάτρα 2007
5. Εισαγωγή στην Ψηφιακή Επεξεργασία Σημάτων
   Δρ. Νικολέττα Νικολάου
6. http://read.pudn.com/downloads50/sourcecode/embed/170149/bsl/tone/tone.c__.htm
7. http://www.survey.ntua.gr/main/courses/general/sigproc/lectures/dsp2005_0809.pdf
8. http://www.iowegian.com/fir/help/fir_design_methods
9. http://www.autom.teithe.gr/gr/akadimaika/dsp/thema3.htm#FIR
10. http://poseidon.csd.auth.gr/New_Courses/signal_processing/oldwebsite/laboratory/index_gr.htm
11. http://autodsp.teipir.gr/Greek_Version/Didaktiko_biblio.htm