**ΕΡΓΑΣΤΗΡΙΟ ΨΗΦΙΑΚΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΣΗΜΑΤΩΝ**

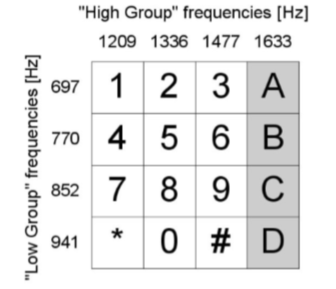
**Εργαστηριακή Άσκηση 7**

**Υλοποίηση DTMF αποκωδικοποιητή**

|  |  |  |  |
| --- | --- | --- | --- |
| **ΟΝΟΜΑ** | **ΕΠΩΝΥΜΟ** | **Α.Μ.** | **ΟΜΑΔΑ** |
|  |  |  |  |
|  |  |  |  |

**Άσκηση 7.1 (υλοποίηση DTMF αποκωδικοποιητή σε περιβάλλον Matlab)**

Σε αυτό το ερώτημα μας ζητάται να υλοποιήσουμε ένα DTMF αποκωδικοποιείται με χρήση του αλγορίθμου Goertzel σε Matlab.Λειτουργεί ως εξής: ανιχνεύει την παρουσία διπλών τόνων ( δηλ.σήμα το οποίο αποτελείται απο υπέρθεση 2 σημάτων με 2 συγκεκριμένες συχνότητες), αποκωδικοποιεί την πληροφορία και τελικά τη μεταφράζει σε ψηφιακή μορφή.Παρακάτω φαίνεται ο πίνακας που μας δίνεται για την αντιστοίχιση των συχνοτήτων στους τόνους.



Για την υλοποίηση :

Αρχικά ορίζουμε 2 πίνακες στους οποίους αρχικοποιούμε τις χαμηλές και τις υψηλές συχνότητες που μας δίνονται, και άλλους 2 στους οποίους τοποθετούμε τις τιμές του συντελεστή , που θα χρησιμοποιήσουμε στον αλγόριθμο Goertzel για κάθε μια από τις συχνότητες αντίστοιχα.Επιπλέον, ορίζουμε στον πίνακα tonos τα πλήκτρα για την αντιστοίχιση και 2 μηδενικούς που θα αποθηκεύσουμε το φασματικό περιεχόμενο που θα προκύψει από τον αλγόριθμο Goertzel.

Ο παρακάτω κώδικας υλοποιείται ολόκληρος για 16 επαναλήψεις ώστε να υπολογιστούν όλοι οι δυνατοί συνδιασμοί μεταξύ των συχνοτήτων αλλά παρουσιάζεται αποσπασματικά.

freqLOW=[697 770 852 941];

freqHIGH=[1209 1336 1477 1633];

coefLOW=[1.703275 1.635585 1.562297 1.482867];

coefHIGH=[1.163138 1.008835 0.790074 0.559454];

tonos=['1','2','3','A';'4','5','6','B';'7','8','9','C';'\*','0','#','D'];

fasmaLow=[0 0 0 0];

fasmaHigh=[0 0 0 0];

Αρχικά δημιουργούμε τα σήματα εισόδου,τα οποία είναι υπέρθεση 2 σημάτων με μια ‘low’ και μια ‘high’ συχνότητα.Αριθμός δειγμάτων : 205, Δειγματοληψία : 8KHz.

for i=1:4

for j=1:4

%sunthetoume olous tous sunduasmous suxnothtwn gia na ftiaksoume ta

%16 shmata,pairnoume 205 deigmata me fs=8000 Hz

X=sin(2\*pi\*(1:205)\*freqLOW(i)/8000)+sin(2\*pi\*(1:205)\*freqHIGH(j)/8000);

Παρακάτω είναι το στάδιο υπολογισμού των φασματικών περιεχομένων με χρήση του αλγορίθμου.

Και ο αλγόριθμος Goertzel υλοποιείται παρακάτω σύμφωνα με τις σχέσεις (10,(14).

function [fasma]=Goer\_algo(input,coeff)

Qn1=[0];

Qn2=[0];

for i = 1:205

Qn=input(i)+coeff\*Qn1-Qn2;

Qn2=Qn1;

Qn1=Qn;

end

Q1=Qn.^2;

Q2=Qn1.^2;

fasma=Q1+Q2-coeff\*Qn\*Qn1;

end

Το αποτέλεσμα που παίρνουμε παρακάτω είναι το πλάτος που αντιστοιχεί σε κάθε μια από τις συχνότητες.Η διαδικασία αυτή γίνεται σε 4 επαναλήψεις,όσες και οι συχνότητες low και high αντίστοιχα.

for k=1:4

%upologizoume to fasmatiko periexomeno gia tis High kai Low suxnotites

fasmaLow(k)=Goer\_algo(X,coefLOW(k));

fasmaHigh(k)=Goer\_algo(X,coefHIGH(k));

Τέλος, ανιχνεύουμε τις μέγιστες συχνότητες που συνθέτουν κάθε τόνο και αντιστοιχίζουμε στο κατάλληλο πλήκτρο.  
  
  
Όταν τρέξουμε το πρόγραμμα μας τυπώνει όλα τα δυνατά πλήκτρα.

o tonos einai o 1

o tonos einai o 2

o tonos einai o 3

o tonos einai o A

o tonos einai o 4

o tonos einai o 5

o tonos einai o 6

o tonos einai o B

%anagnwrish tonou:vriskoume tis megistes sunistwses kai

%antistoixizoume me ton pinaka tonos

[frH,posH]=max(fasmaHigh);

[frL,posL]=max(fasmaLow);

disp(sprintf(('o tonos einai o %c\n'),tonos(posL,posH)));

end

end

o tonos einai o 7

o tonos einai o 8

o tonos einai o 9

o tonos einai o C

o tonos einai o \*

o tonos einai o 0

o tonos einai o #

o tonos einai o D

**Άσκηση 7.2 (υλοποίηση DTMF αποκωδικοποιητή σε περιβάλλον TMS320C6713)**

Στην παρούσα αναφορά θα περιγράψουμε τα ‘κομμάτια΄ του κώδικα που διαφέρουν απο τις προηγούμενες ασκήσεις.Για την υλοποίηση επιλέξαμε να χρησιμοποιήσουμε interrupts.Η ανίχνευση των τόνων πραγματοποιείται στα πλαίσια της ISR.

Οι συναρτήσεις που χρησιμοποιούμε είναι:

**short** **int** **Q15**(**short** **int** a,**short** **int** b,**int** shift){

**int** temp;

**short** **int** Q;

//me to orisma shift diaxwrizoume tous pollaplasiasmous pou den xreiazontai to X2

temp=(**int**)a\*(**int**)b;

**if** (shift==0)

Q=(**short**)(temp>>15);

**else**

Q=(**short**)(temp>>14);

**return** Q;

}

Με την συνάρτηση Q15 γίνονται οι πολ/μοι οπως και στην προηγούμενη άσκηση. Το όρισμα shift χρησιμοποιείται για να διαχωρίσουμε τις περιπτώσεις των συντελεστών που διαιρέθηκαν με το 2 πριν την αναπαρασταση τους σε Q15 format. Δίνοντας ως όρισμα το 0 η συνάρτηση πολ/ζει τους 16 bit αριθμούς και κάνει shift right 15 για να επαναφέρει το αποτέλεσμα σε Q15 αναπαράσταση. Για όρισμα διαφορετικό του 0 η συνάρτηση κανει shift right 14 στο αποτέλεσμα ενσωματώνοντας έτσι τη διαίρεση με το 2.

**short** **GoertzelAlgorithm**(**short** coeff,**short** \*input,**int** flag){

**int** i;

**short** Y,temp,Q[3],temp1,temp2;

Q[1]=0;

Q[2]=0;

**for**(i=0;i<205;i++){

Q[0]=input[i]+Q15(coeff,Q[1],flag)-Q[2];

Q[2]=Q[1];

Q[1]=Q[0];

}

temp=Q15(coeff,Q[0],flag);

temp1=Q15(Q[0],Q[0],0);

temp2=Q15(Q[1],Q[1],0);

Y=temp1+temp2+Q15(Q[1],temp,0);

**return** Y;

}

Η συνάρτηση **GoertzelAlgorithm** δέχεται ως όρισμα έναν buffer 205 στοιχείων που χρειάζονται για τον υπολογισμό του πλάτους της εκάστοτε συχνότητας. Τον συντελεστή coeff της αντίστοιχης συχνότητας και μια μεταβλητη flag ώστε να γνωρίζουμε αν ο συντελεστής διαιρέθηκε με το 2.Υπολογίζουμε για 205 επαναλήψεις το feedback κομματι του αλγορίθμου και τελος στην μεταβλητή Υ μια φορά το feedforward κομμάτι που μας δίνει το τελικό αποτέλεσμα.Παρατηρούμε οτι η μεταβλητή flag χρησιμοποιείται μονο για τους πολ/μους με συντελεστή(coeff) ενω για τους υπόλοιπους δίνουμε ως όρισμα το 0.

**int** **findmax**(**short** \*buffer){

**int** i,posmax=0;

**short** max=buffer[0];

**for**(i=1;i<4;i++){

**if**(buffer[i]>max){

max=buffer[i];

posmax=i;

}

}

**return** posmax;

}

Η συνάρτηση findmax δέχεται ως όρισμα ένα buffer 4 στοιχείων και επιστρέφει τη θέση στην οποία βρίσκεται το μεγαλύτερο στοιχείο του.

Στη συνέχεια ορίζουμε(global) τις μεταβλητές και τους buffer που θα χρησιμοποιηθούν.

**short** buffer[205];

**int** f\_low[4] = {697, 770, 852, 941}, f\_high[4] = {1209, 1336, 1477,1633};

**short** coeff\_low[4]={0x6D02,0x68AD,0x63FC,0x5EE7},coeff\_high[4]={0x4A70,0x4090,0x6521,0x479C};

**const** **char** keys[4][4] = {{'1', '2', '3', 'A'}, {'4', '5', '6', 'B'}, {'7', '8','9', 'C'}, {'\*', '0', '#', 'D'}};

**int** N=0,prevLow=100,prevHigh=100;

Ο πίνακας buffer χρησιμοποιείται για την αποθήκευση των 205 δειγμάτων του σήματος. Οι f\_low ,f\_high περιέχουν τις high και low συχνότητες. Οι coeff\_low, coeff\_high περιέχουν τους συντελεστές των αντίστοιχων συχνοτήτων σε Q15 format. Ο πίνακας keys περιέχει τους χαρακτήρες των τόνων.Η μεταβλητή N μετρά πόσα δείγματα εχουμε αποθηκεύσει ενώ οι prevLow,prevHigh που αρχικοποιούνται στην τιμή 100 χρησιμοποιούνται για να ελέγξουμε αν άλλαξε ο τόνος και αντίστοιχα να τυπώσουμε την αλλαγή.

**void** **main**(**void**){

**DSK6713\_init**(); // Initialize the board support library, must be called first

hCodec = **DSK6713\_AIC23\_openCodec**(0, &config); // open codec and get handle

// set codec sampling frequency 48kHz

**DSK6713\_AIC23\_setFreq**(hCodec, DSK6713\_AIC23\_FREQ\_8KHZ); //allazoume tis Fs apo 48 se 8 KHz

\*(**unsigned** **volatile** **int** \*)McBSP1\_RCR = 0x00A0;

\*(**unsigned** **volatile** **int** \*)McBSP1\_XCR = 0x00A0;

// comm\_intr();

init\_hw\_interrupts();

**while**(1);

}

Το κομμάτι της main παραμένει ίδιο με τα προηγούμενα με μόνη διαφορά οτι ορίζουμε συχνότητα δειγματοληψίας 8kHz.

Αρχικά αποθηκεύμε την είσοδο στην μεταβλητή data με την εντολή input\_leftright\_sample(); και στη συνέχεια κάνουμε cast στην μεταβλητή short input για να απομονώσουμε το ένα κανάλι.Στη συνέχεια παρατηρήσαμε οτι δεχόμασταν interrupts χωρίς να δίνουμε καποια είσοδο και σαν input λαμβάναμε τιμές στο [-1,1].Με την **if** (input>2 || input<-2) αποφεύγουμε αυτες τις τιμές. Η **if**(N<205) εξασφαλίζει οτι περιμένουμε να αποθηκεύσουμε 205 δείγματα πρίν αρχίσουμε την επεξεργασία.Η μεταβλητη flag αρχικοποιείται με την τιμή 1 και αλλάζει μονο στη περίπτωση των συχνοτήτων 1477,1633 Hz που δεν διαιρέθηκαν με το 2.

**interrupt** **void** **serial\_port\_rcv\_isr**(){

**int** data,i,posLow,posHigh,flag=1,dif\_Low,dif\_High;

**short** input,fasmaLow[4],fasmaHigh[4];

//lamvanw 32 bit eisodo kai kanw cast se 16 bit gia na apomonwsw to ena kanali

data=input\_leftright\_sample();

input=(**short**)(data>>16);

**if** (input>2 || input<-2){

//apothikeuw ta 205 deigmata tou shmatos pou xreiazetai o algorithmos

**if**(N<205)

buffer[N++]=input;

**if**(N==205){

**for**(i=0;i<4;i++){

**if**(i>1)

flag=0;

fasmaHigh[i]=GoertzelAlgorithm(coeff\_high[i],buffer,flag);

fasmaLow[i]=GoertzelAlgorithm(coeff\_low[i],buffer,1);

}

Μόλις αποθηκεύσουμε τα 205 δείγματα που χρειάζεται ο αλγόριθμος υπολογίζουμε σε δυο πίνακες των 4 στοιχείων (fasmaHigh[4],fasmaLow[4]) χωριστά τα πλάτη των high και low συνιστωσών ωστε να διευκολύνουμε την εύρεση του μεγίστου.Οι low συχνότητες εχουν διαιρεθεί ολές με το 2 γι’αυτό δίνουμε το 1 ως όρισμα στη συνάρτηση.Για τις high χρησιμοποιούμε το όρισμα flag =1 που παίρνει την τιμή 0 για τις 2 μεγαλύτερες συχνότητες που δεν διαιρέθηκαν με το 2.

Υπολογίζουμε στις posHigh,posLow μέσω της συνάρτησης findmax τις μέγιστες (high,low) συνιστώσες του τόνου και στη συνέχεια τις αφαιρούμε απο τις prevHigh,prevLow για να ελέγξουμε αν ο τόνος άλλαξε και αντίστοιχα να τυπώσουμε την αλλαγή.Στην συνέχεια αποθηκεύουμε τα τωρινά μέγιστα ως προηγούμενα για να συνεχίσουμε τον ελεγχο.Μηδενίζουμε το Ν για να μετρήσουμε πάλι 205 δείγματα, και στέλνουμε την είσοδο στην έξοδο για να τερματίσει η ISR.

posLow=findmax(fasmaLow);

posHigh=findmax(fasmaHigh);

if\_Low=prevLow-posLow;

dif\_High=prevHigh-posHigh;

**if**(dif\_High!=0 || dif\_Low!=0)

**printf**("Low freq=%d High freq=%d\n key=%c\n",f\_low[posLow],f\_high[posHigh],keys[posLow][posHigh]);

prevLow=posLow;

prevHigh=posHigh;

N=0;

output\_leftright\_sample(data);

}

}

**return**;

}