ΕΡΓΑΣΤΗΡΙΑΚΗ 3 ΑΣΚΗΣΗ

Σύστημα Σηματοδότησης

Dual Tone Multi Frequency

1. Εισαγωγή

Στα πλαίσια αυτής της άσκησης θα υλοποιηθεί στην αναπτυξιακή κάρτα TMS320C6711 DSK το σύστημα σηματοδότησης Dual Tone Multi Frequency.

2. Το σύστημα σηματοδότησης Dual Tone Multi Frequency

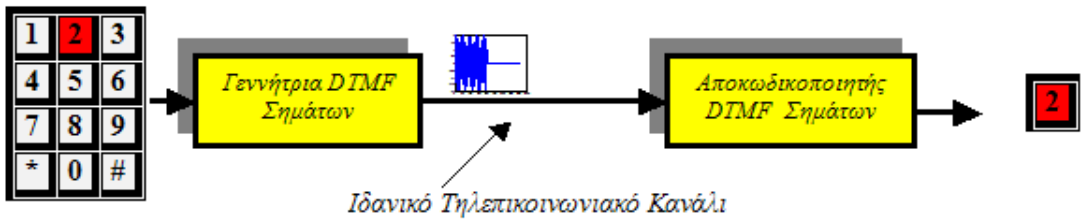
Το συγκεκριμένο σύστημα σηματοδότησης χρησιμοποιείται ευρέως στα συστήματα τηλεφωνικής κλήσης. Το πάτημα ενός πλήκτρου μίας τηλεφωνικής συσκευής έχει σαν αποτέλεσμα την δημιουργία ενός ***DTMF*** (***D***ual ***T***one ***M***ulti ***F***requency) σήματος σύμφωνα με τον παρακάτω πίνακα.

DTMF Κωδικοποίηση			
Συχνότητα	1209 Hz	1336 Hz	1477 Hz
697 Hz	**1**	**2**	**3**
770 Hz	**4**	**5**	**6**
852 Hz	**7**	**8**	**9**
941 Hz	*****	**0**	**#**

Πίνακας 1.

Οι συχνότητες που περιέχονται στον Πίνακα 1 είναι επιλεγμένες έτσι ώστε καμιά από αυτές να μην είναι πολλαπλάσιο κάποιας άλλης καθώς επίσης το άθροισμα και η διαφορά οποιωνδήποτε από τις παραπάνω συχνότητες να μην παράγει κάποια από τις άλλες συχνότητες που εμφανίζονται στον παραπάνω πίνακα.

Στο Σχηματικό διάγραμμα που ακολουθεί φαίνεται το συνολικό σύστημα το οποίο θα πρέπει να υλοποιηθεί στα πλαίσια αυτής της εργασίας. Παρατηρήστε ότι αρχικά το τηλεπικοινωνιακό κανάλι θεωρούμε ότι είναι ιδανικό δηλαδή η απόκριση συχνότητάς του έχει μέτρο ίσο με την μονάδα και δεν υπάρχει παρουσία θορύβου. Αυτό βέβαια στην πραγματικότητα δεν ισχύει γι' αυτό και σε επόμενη ενότητα θα πειραματιστείτε και για κανάλι με παρουσία προσθετικού Γκαουσιανού θορύβου.



Σχήμα 1.

Κατά την φάση της δημιουργίας ενός DTMF σήματος, καλούμαστε να βρούμε ένα σύστημα το οποίο σε κάθε πάτημα ενός ψηφίου ή συμβόλου της τηλεφωνικής συσκευής, θα δημιουργεί στην έξοδό του ένα σήμα συγκεκριμένης χρονικής διάρκειας το οποίο θα προκύπτει από την υπέρθεση δύο σημάτων των οποίων οι συχνότητες αντιστοιχούν στο συγκεκριμένο ψηφίο (σύμφωνα με τον Πίνακα 1) της τηλεφωνικής συσκευής που πατήσαμε. Αν δηλαδή για παράδειγμα πατήσουμε το πλήκτρο με το ψηφίο 2 τότε το σύστημα θα πρέπει να δημιουργήσει στην έξοδό του το ακόλουθο σήμα

$$x(t)=cos(2\pi 697t) + cos(2\pi 1336t)$$

Σημειώνεται ότι για να είναι εφικτή η σωστή αποκωδικοποίηση ενός DTMF σήματος, στα χρησιμοποιούμενα τηλεφωνικά συστήματα παρεμβάλλεται ένα σήμα «σιωπής» (x(t)=0).

Ερώτημα 1: Καλείστε να υλοποιήσετε μια γεννήτρια η οποία να παράγει τα DTMF σήματα που αντιστοιχούν στα ψηφία της τηλεφωνικής συσκευής. Η διάρκεια κάθε ψηφίου θα πρέπει να είναι ανάμεσα στα 40-44msec. Θεωρήστε ότι κάθε ψηφίο θα συνοδεύεται από σήμα «σιωπής» (x(t)=0), διάρκειας 20msec. Ως συχνότητα δειγματοληψίας θεωρήστε την τιμή *fs* = 8000Hz.

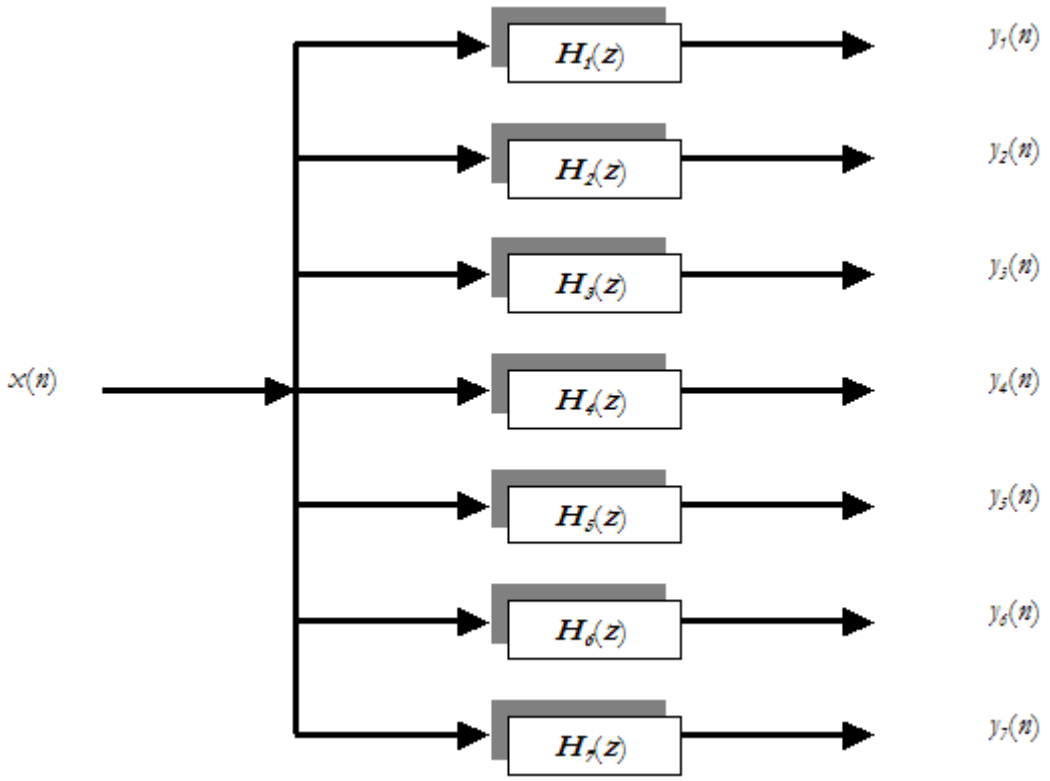
Κατά την φάση της αποκωδικοποίησης (decoding) ενός DTMF σήματος το τηλεφωνικό κέντρο καλείται να αποφασίσει σε ποιο ψηφίο ή σύμβολο αντιστοιχεί το DTMF σήμα το οποίο έλαβε.

Έχουν προταθεί στην βιβλιογραφία αρκετοί τρόποι για την υλοποίηση του αποκωδικοποιητή. Στην εργασία αυτή ζητείται η υλοποίηση των παρακάτω δύο (2) διαφορετικών τρόπων.

- *Υπολογίζοντας τον Διακριτό Μετασχηματισμό Fourier (DFT) του DTMF σήματος.*
  Η ιδέα είναι απλή, αφού το DTMF σήμα συντίθεται από δύο ημιτονικά σήματα περιμένουμε το ενεργειακό του περιεχόμενό του να είναι

συγκεντρωμένο γύρω από δύο συγκεκριμένες συχνότητες. Για τον υπολογισμό του DFT μπορούμε να χρησιμοποιήσουμε είτε FFT είτε τον αλγόριθμο του Goertzel.

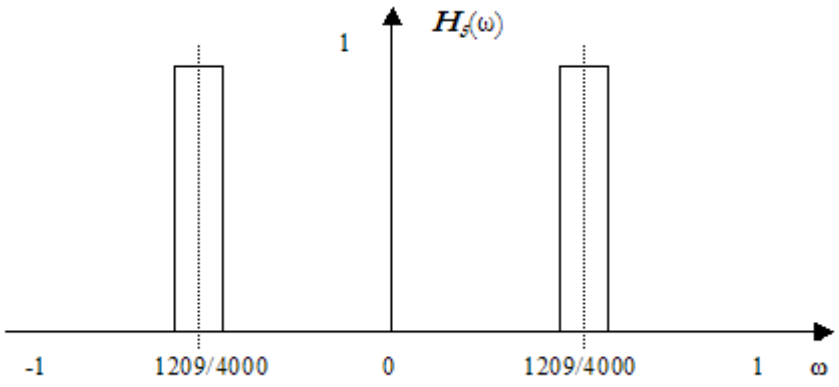
- *Χρησιμοποιώντας Συστοιχία Φίλτρων (Filter Bank).*
  Ένας άλλος τρόπος με τον οποίο μπορούμε να υλοποιήσουμε τον αποκωδικοποιητή είναι να περάσουμε το DTMF σήμα μέσα από μία συστοιχία φίλτρων (Filter Bank) όπως αυτή που φαίνεται στο Σχήμα 2.



Σχήμα 2.

Παρατηρήστε ότι κάθε φίλτρο πρέπει να σχεδιαστεί έτσι ώστε να περνάει μόνο μία συχνότητα από τις επτά που υπάρχουν στον Πίνακα 1. Αν για παράδειγμα υποθέσουμε ότι το πρώτο φίλτρο της συστοιχίας επιτρέπει την διέλευση της μικρότερης συχνότητας (697 Hz), που εμφανίζεται στον Πίνακα 1, και το έβδομο φίλτρο της συστοιχίας επιτρέπει την διέλευση της μεγαλύτερης συχνότητας (1477 Hz), που εμφανίζεται στον ίδιο πίνακα, η απόκριση συχνότητας του πέμπτου φίλτρου της συστοιχίας $\boldsymbol{H_5(\omega)}$ θα πρέπει να προσεγγίζει την ιδανική απόκριση συχνότητας που φαίνεται στο παρακάτω σχήμα.

Για να αποφασίσουμε τέλος το ψηφίο ή το σύμβολο που αντιστοιχεί στο DTMF σήμα που οδηγήσαμε στην είσοδο της συστοιχίας φίλτρων, υπολογίζουμε την ενέργεια των σημάτων $y_i(n)$ i=1,2,..,7 των εξόδων της συστοιχίας φίλτρων και απομονώνουμε τις δύο εκείνες εξόδους που έχουν την μεγαλύτερη ενέργεια.



Σχήμα 3: Ιδανική απόκριση συχνότητας του πέμπτου φίλτρου της συστοιχίας.

Για την προσέγγιση των ιδανικών προδιαγραφών που φαίνονται στο Σχήμα 3, μπορούμε να χρησιμοποιήσουμε είτε:

Κ1 Ζωνοδιαβατό (Bandpass) FIR φίλτρο (προαιρετικό)
Κ2 FIR φίλτρο Εγκοπής (προαιρετικό)
Κ3 IIR φίλτρο Εγκοπής

3. Το πρόγραμμα codec_edma.c

Ακολουθώντας την πρακτική της πρώτης άσκησης, σας δίνεται ένα πρόγραμμα που υλοποιεί ένα σύστημα δειγματοληψίας και ανακατασκευής ενός ακουστικού σήματος. Η βασική διαφορά του προγράμματος αυτού με το loopback.c έγκειται στο ότι, πλέον, η επεξεργασία των δεδομένων γίνεται ανά μπλοκ κι όχι ανά δείγμα. Για το σκοπό αυτό θα χρησιμοποιηθεί ο ελεγκτής της μονάδας EDMA. Παρακάτω θα αναλυθούν τα τμήματα κώδικα του codec_edma.c που το διαφοροποιούν από το loopback.c και στην ουσία αρχικοποιούν κατάλληλα τον ελεγκτή της μονάδας EDMA.

Μόλις ληφθεί ένα δείγμα από τη συσκευή AD535, δηλαδή μόλις ο DRR της McBSP0 λάβει και τα 16 bit δείγματος, παράγεται το σήμα RINT0. Με παρόμοια διαδικασία μόλις αδειάσει ο DXR και είναι έτοιμος, πλέον, να γεμίσει με το επόμενο δείγμα, παράγεται το σήμα XINT0. Στη συγκεκριμένη υλοποίηση, δε θέλουμε τα σήματα αυτά να γίνουν αντιληπτά από τη CPU, όπως στις δύο πρώτες ασκήσεις, αλλά να χρησιμοποιηθούν ως γεγονότα συγχρονισμού για τις EDMA μεταφορές, ενεργοποιώντας τα αντίστοιχα κανάλια.

Κάθε φορά που θα γεννιέται το σήμα RINT0, ο ελεγκτής EDMA θα αναλαμβάνει τη μεταφορά του δείγματος από το DRR σε μια προκαθορισμένη θέση μνήμης, αντιστοίχως όταν γεννιέται το XINT0, ο ελεγκτής EDMA θα αναλαμβάνει τη μεταφορά του δείγματος από μια προκαθορισμένη θέση μνήμης στο DXR. Τα χαρακτηριστικά των EDMA μεταφορών καθορίζονται από το χρήστη μέσω των πεδίων της PaRAM κι αναλύονται παρακάτω.

Μόλις πραγματοποιηθεί το καθορισμένο, από τα πεδία της PaRAM, πλήθος μεταφορών, ο ελεγκτής της EDMA παράγει το σήμα διακοπής EDMA_INT προς στη CPU, έτσι ώστε να εκτελεστεί η αντίστοιχη ρουτίνα εξυπηρέτησης διακοπής (ISR). Όπως, γίνεται αντιληπτό, η

CPU ειδοποιείται, μέσω αυτού του σήματος διακοπής, ανά μπλοκ δειγμάτων κι όχι ανά δείγμα, όπως συνέβαινε στις δύο πρώτες ασκήσεις.

Το σήμα διακοπής EDMA_INT είναι by default συνδεδεμένο με το INT8 της CPU. Επισημαίνεται ότι για το λόγο αυτό δε χρειάζεται να αρχικοποιηθούν τα πεδία των καταχωρητών IMH και IML. Το INT8 ενεργοποιείται μέσω των ακόλουθων εντολών :

```
ICR = 0xffff;
IER |= 0x102;
CSR |= 1;
```

Στη συνέχεια καλείται η συνάρτηση αρχικοποίησης της EDMA ***edma_init()***. Στις πρώτες γραμμές της συνάρτησης, προσδιορίζονται οι τιμές των παρακάτω τεσσάρων καταχωρητών:

```
*(unsigned volatile int *)ECR = 0xffff;
*(unsigned volatile int *)EER = 0x3000;
*(unsigned volatile int *)CIPR = 0xffff;
*(unsigned volatile int *)CIER = 0x100;
```

Ο **E**nable **C**lear **R**egister (ECR) παίρνει την τιμή 0xffff. Έτσι «καθαρίζονται» και τα 16 events, όπου το κάθε bit αποκτώντας την τιμή 1 καθαρίζει και το αντίστοιχο event που τυχόν να ήταν ενεργό στον αντίστοιχο καταχωρητή γεγονότων (**E**vent **R**egister ER). Η CPU χρησιμοποιεί τον ECR για να αποφύγει τις όποιες λανθασμένες συνθήκες μπορεί να προκύψουν κατά την αρχικοποίηση γύρω από την κατάσταση κάποιου event.

Μέσω του **E**vent **E**nable **R**egister (EER) ο EDMA controller επιτρέπει στο χρήστη να επιλέξει ποια events θα ενεργοποιηθούν. Στη προκειμένη περίπτωση με την τιμή 0x3000 ενεργοποιούνται τα events 12 και 13, αφού τα αντίστοιχα bits στο καταχωρητή αυτό λαμβάνουν τη τιμή 1. Τα γεγονότα αυτά έχουν ακρωνύμια XEVT0 και REVT0 και αποτελούν σήματα αποστολής και λήψης (XINT0 και RINT0 αντίστοιχα) που προέρχονται από το McBSP0.

Ο **C**hannel **I**nterrupt **P**ending **R**egister (CIPR) με την τιμή 0xffff καθαρίζει και τα 16 interrupts. Κάθε φορά η νέα τιμή που θα αποκτήσει ο CIPR, καθορίζεται από την προγραμματιζόμενη τιμή που έχει το **T**ransfer **C**omplete **C**ode (TCC) πεδίο της παραμέτρου OPT ενός γεγονότος, και τη λαμβάνει μετά την ολοκλήρωση μιας μεταφοράς δεδομένων στο αντίστοιχο κανάλι του γεγονότος αυτού. Καθένα από τα interrupts αυτά τα χρησιμοποιεί ο EDMA controller για να παράγει ένα γενικό EDMA_INT προς τη CPU, εφόσον βέβαια έχει ολοκληρωθεί, όπως αναφέρθηκε, μία μεταφορά δεδομένων.

Για να παραχθεί όμως το EDMA_INT, θα πρέπει επιπλέον να έχει τεθεί το αντίστοιχο bit ενεργοποίησης διακοπής στον **C**hannel **I**nterrupt **E**nable **R**egister (CIER). Γι’ αυτό και στη

συνέχεια ο CIER παίρνει την τιμή 0x100, έτσι ώστε τελικά να διατηρείται ενεργοποιημένο το **C**hannel **I**nterrupt **E**nable 8 (CIE8) και να μπορεί να συνδυαστεί στη συνέχεια με την αντίστοιχη τιμή του interrupt στον CIPR (CIP8) για την παραγωγή του EDMA_INT. Έτσι, το EDMA_INT που στέλνεται κάθε φορά από τον EDMA controller στη CPU θα επιτρέψει τελικά στην τελευταία να εκτελέσει μία συγκεκριμένη interrupt service ρουτίνα (ISR). Στη ρουτίνα αυτή θα πρέπει να καθαριστεί αρχικά το αντίστοιχο bit στον CIPR έτσι ώστε να μπορεί η CPU να αναγνωρίσει και άλλες επιπλέον διακοπές. Αυτή η ενέργεια της CPU θα περιγραφεί περισσότερο κατά την ανάλυση της ISR που θα δοθεί παρακάτω.

Επόμενο βήμα στον κώδικα, αποτελεί ο προσδιορισμός των παραμέτρων για τα γεγονότα C (12) και D (13) του ελεγκτή της EDMA:

```
  *(unsigned volatile int *)(EVENTC_PARAMS+OPT) = 0x49000002;
  *(unsigned volatile int *)(EVENTC_PARAMS+SRC) = pcm_out;
  *(unsigned volatile int *)(EVENTC_PARAMS+CNT) = BLOCK_SIZE;
  *(unsigned volatile int *)(EVENTC_PARAMS+DST) = McBSP0_DXR;
  *(unsigned volatile int *)(EVENTC_PARAMS+IDX) = 0;
  *(unsigned volatile int *)(EVENTC_PARAMS+LNK) =
EVENTN_PARAMS;

  *(unsigned volatile int *)(EVENTD_PARAMS+OPT) = 0x48380002;
  *(unsigned volatile int *)(EVENTD_PARAMS+SRC) = McBSP0_DRR;
  *(unsigned volatile int *)(EVENTD_PARAMS+CNT) = BLOCK_SIZE;
  *(unsigned volatile int *)(EVENTD_PARAMS+DST) = pcm_in;
  *(unsigned volatile int *)(EVENTD_PARAMS+IDX) = 0;
  *(unsigned volatile int *)(EVENTD_PARAMS+LNK) =
EVENTO_PARAMS;
```

Εδώ καθορίζονται οι παραμέτροι για τα συγκεκριμένα events του EDMA. Συγκεκριμένα, καθορίζεται η τιμή των πεδίων Options, SRC Address, Array/Frame Count – Element Count, DST Address, Array/Frame Index – Element Index και Element Count Reload – Link Address. Οι πρώτες έξι εντολές καθορίζουν τις παραμέτρους για το κανάλι 12. Στη πρώτη εντολή εγγράφεται στη παράμετρο Options η τιμή 0x49000002 με την οποία τα πεδία της λαμβάνουν τις ακόλουθες τιμές:

Πεδίο	**Περιγραφή**
FS = 0	Το κανάλι συγχρονίζεται κατά element/array
LINK = 1	Ενεργοποιείται η διαδικασία linking για τις παραμέτρους του event. Όταν το παρόν σύνολο των παραμέτρων εξαντλείται, οι εγγραφές για το συγκεκριμένο κανάλι επαναπροσδιορίζονται με βάση τις παραμέτρους που καθορίζονται από τη link διεύθυνση
TCC = 0000	Ο κωδικός αυτός ολοκλήρωσης μεταφοράς θέτει το αντίστοιχο bit στον CIPR

	όταν το TCINT = 1 και το παρόν σύνολο παραμέτρων έχει εξαντληθεί
TCINT = 0	Είναι απενεργοποιημένη η ένδειξη ολοκλήρωσης μεταφοράς
DUM = 00	Σταθερή διεύθυνση προορισμού
2DD = 0	Μονοδιάστατος προορισμός
SUM = 01	Η αύξηση της διεύθυνσης πηγής εξαρτάται από τα πεδία 2DS και FS
2DS = 0	Μονοδιάστατη πηγή
ESIZE = 01	Το μέγεθος του element είναι 16-bit half-word
PRI = 010	Χαμηλή προτεραιότητα για τις EDMA μεταφορές

Πίνακας 2.

Η επόμενη παράμετρος που προσδιορίζεται είναι η SRC Address στην οποία αποθηκεύεται η αρχική διεύθυνση του buffer στον οποίο θα μεταφερθεί το block δεδομένων, και που ουσιαστικά στη περίπτωση μας αποτελεί τη πρώτη θέση του πίνακα στον οποίον αποθηκεύονται τα δεδομένα που θέλουμε να οδηγηθούν στην έξοδο της κάρτας. Ακολούθως προσδιορίζεται η τιμή του Element Count ώστε αυτή να είναι ίση με τη τιμή της σταθεράς BLOCK_SIZE. Δηλαδή το μέγεθος του κάθε block μετάδοσης είναι BLOCK_SIZE τιμές των 16-bit. Στη παράμετρο DST Address δίνεται η διεύθυνση του **D**ata **T**ransmit **R**egister (DXR) για το McBSP0, όπου και θέλουμε να καταλήξουν τα δεδομένα που πάρθηκαν από την SRC διεύθυνση. Στη συνέχεια αποθηκεύεται η τιμή 0 στη παράμετρο Element Index θέτοντας offset 0, προκειμένου να μεταφερθούν δεδομένα που βρίσκονται σε διαδοχικές θέσεις μνήμης. Τελευταία ενέργεια στο προσδιορισμό των παραμέτρων για το event C, είναι η αποθήκευση της διεύθυνσης EVENTN_PARAMS από όπου θα γίνεται η επαναφόρτωση (reload) των παραμέτρων για τις επόμενες μεταφορές του καναλιού (εφόσον έχουμε LINK = 1).

Παρακάτω πρόκειται να ακολουθηθεί αντίστοιχη διαδικασία ανάλυσης – επεξήγησης για το event D. Έτσι έχουμε για τη παράμετρο Options του γεγονότος αυτού:

Πεδίο	**Περιγραφή**
FS = 0	Το κανάλι συγχρονίζεται κατά element/array
LINK = 1	Ενεργοποιείται η διαδικασία linking για τις παραμέτρους του event. Όταν το παρόν σύνολο των παραμέτρων εξαντλείται, οι εγγραφές για το συγκεκριμένο κανάλι επαναπροσδιορίζονται με βάση τις παραμέτρους που καθορίζονται από τη link διεύθυνση
TCC = 1000	Ο κωδικός αυτός ολοκλήρωσης μεταφοράς θέτει το αντίστοιχο bit στον CIPR όταν το TCINT = 1 και το παρόν σύνολο παραμέτρων έχει εξαντληθεί. Εδώ τίθεται το bit 8 στον CIPR όταν εξαντλείται το σύνολο παραμέτρων
TCINT = 1	Το bit 8 του CIPR τίθεται μετά την ολοκλήρωση μεταφοράς στο κανάλι.
DUM = 00	Η αύξηση της διεύθυνσης προορισμού εξαρτάται από τα πεδία 2DS και FS
2DD = 0	Μονοδιάστατος προορισμός
SUM = 01	Σταθερή διεύθυνση πηγής

2DS = 0	Μονοδιάστατη πηγή
ESIZE = 01	Το μέγεθος του element είναι 16-bit half-word
PRI = 010	Χαμηλή προτεραιότητα για τις EDMA μεταφορές

Πίνακας 3.

Η παράμετρος SRC Address για το event 13, είναι η διεύθυνση του καταχωρητή **D**ata **R**eceive **R**egister (DRR) από τον οποίο διαβάζει τα δεδομένα εισόδου ο EDMA Controller. Οι παράμετροι Element Count και Element Index παραμένουν ίδιες όπως και στο event 12 και είναι ίσες με BLOCK_SIZE και 0 αντίστοιχα. Στη παράμετρο DST Address αποθηκεύεται η διεύθυνση της πρώτης θέση του πίνακα στον οποίον αποθηκεύονται τα δεδομένα εισόδου της κάρτας. Τέλος, αφού έχουμε πάλι LINK = 1, η επαναφόρτωση (reload) των παραμέτρων για τις επόμενες μεταφορές του καναλιού θα γίνεται από τη διεύθυνση EVENTO_ PARAMS.

Οι παράμετροι επαναφόρτωσης (reload parameters) για το κανάλι C είναι:

```
  *(unsigned volatile int *)(EVENTN_PARAMS+OPT) = 0x49000002;
  *(unsigned volatile int *)(EVENTN_PARAMS+SRC) = src;
  *(unsigned volatile int *)(EVENTN_PARAMS+CNT) = BLOCK_SIZE;
  *(unsigned volatile int *)(EVENTN_PARAMS+DST) = McBSP0_DXR;
  *(unsigned volatile int *)(EVENTN_PARAMS+IDX) = 0;
  *(unsigned volatile int *)(EVENTN_PARAMS+LNK) =
EVENTN_PARAMS;
```

και για το κανάλι D είναι:

```
  *(unsigned volatile int *)(EVENTO_PARAMS+OPT) = 0x48380002;
  *(unsigned volatile int *)(EVENTO_PARAMS+SRC) = McBSP0_DRR;
  *(unsigned volatile int *)(EVENTO_PARAMS+CNT) = BLOCK_SIZE;
  *(unsigned volatile int *)(EVENTO_PARAMS+DST) = dst;
  *(unsigned volatile int *)(EVENTO_PARAMS+IDX) = 0;
  *(unsigned volatile int *)(EVENTO_PARAMS+LNK) =
EVENTO_PARAMS;
```

Επισημαίνεται ότι στη σύνδεση των μεταφορών, οι παράμετροι όλων των μεταφορών εκτός από τη πρώτη καθορίζονται από τις παραμέτρους επαναφόρτωσης του αντίστοιχου καναλιού. Συγκεκριμένα, για το κανάλι 12 (C) η μεταφορά του πρώτου μπλοκ θα γίνει με βάση τις παραμέτρους που ξεκινάνε στη διεύθυνση EVENTC_ PARAMS ενώ όλες οι άλλες μεταφορές καθορίζονται από τις παραμέτρους επαναφόρτωσης που ξεκινούν από τη διεύθυνση EVENTN_ PARAMS. Βλέπουμε ότι η μόνη διαφορά έγκειται στην τιμή της παραμέτρου SRC Ως αποτέλεσμα έχει, η αρχική διεύθυνση του πρώτου μπλοκ δεδομένων που θα σταλεί στον DXR να καθορίζεται από την pcm_out, ενώ οι αρχικές διευθύνσεις όλων

των υπολοίπων μπλοκ να καθορίζονται από την τιμή της μεταβλητής src, που ανανεώνεται κάθε φορά που εκτελείται η ISR. Η ίδια ακριβώς λογική ακολουθείται και για το κανάλι 13 (D).

Ακολούθως θα αναφερθούμε στην ISR, που όπως ειπώθηκε πιο πάνω καλείται όταν ο EDMA controller παράγει ένα γενικό EDMA_INT προς τη CPU κατά την ολοκλήρωση μίας μεταφοράς δεδομένων. Ειδικότερα όταν γεμίζει ένα μπλοκ τότε παράγει ο EDMA controller το κατάλληλο interrupt στη CPU. Η CPU εξυπηρετεί τη συγκεκριμένη διακοπή ενώ ο EDMA συνεχίζει να γεμίζει το επόμενο μπλοκ με νέες τιμές που λαμβάνει από τον A/D . Για το λόγο αυτό, το μέγεθος των πινάκων in, out είναι NUM_OF_BLOCKS*BLOCK_SIZE όπου NUM_OF_BLOCKS > 2. Όταν ο EDMA ολοκληρώσει τη συμπλήρωση του τελευταίου στη σειρά μπλοκ, ξεκινά και πάλι να συμπληρώνει το πρώτο μπλοκ με δεδομένα. Αυτή η κυκλική διαδικασία συνεχίζεται για όσο διάστημα μεταφέρονται τιμές από και προς τον A/D. Ο κώδικας που υλοποιεί αυτή την κυκλική ανανέωση παρατίθεται εδώ:

```
if ( dma_index  == NUM_OF_BLOCKS -1)
{
    dma_index =0;
    dst = pcm_in;
    src =  pcm_out;
}
else
{
    dma_index++;
    src += BLOCK_SIZE*DATA_SIZE;
    dst += BLOCK_SIZE*DATA_SIZE;
}
```

Στο παραπάνω κώδικα το νέο μπλοκ δεδομένων που θα διαβάσει ο EDMA controller καθορίζεται από τις νέες τιμές που παίρνουν οι διευθύνσεις src και dst. Στη συνέχεια θα πρέπει να καθαριστεί και το αντίστοιχο bit στον CIPR, όπως αναφέρθηκε. Ύστερα, επαναπροσδιορίζονται οι αρχικές διευθύνσεις για τις παραμέτρους του καναλιού 12 και 13. Αυτές παίρνουν ως τιμές τις νέες διευθύνσεις των src και dst. Έτσι, στο πεδίο SRC Address των παραμέτρων επαναφόρτωσης του καναλιού 12 αποθηκεύεται η αρχική διεύθυνση για το επόμενο block δεδομένων που πρόκειται να οδηγηθεί στην έξοδο της κάρτας και στο πεδίο DST Address των παραμέτρων επαναφόρτωσης του καναλιού 13 αποθηκεύεται η αρχική διεύθυνση για το επόμενο block δεδομένων εισόδου στην κάρτα:

```
*(unsigned volatile int *)CIPR |= 0x10;
*(unsigned volatile int *)(EVENTN_PARAMS+SRC) = src;
*(unsigned volatile int *)(EVENTO_PARAMS+DST) = dst;
```

Με τις παρακάτω γραμμές κώδικα ελέγχουμε πότε έχουμε φτάσει στο τελευταίο μπλοκ των πινάκων, έτσι ώστε οι επόμενες μεταφορές να ξεκινήσουν πάλι από την αρχή.

```
temp = (int)in_ptr - pcm_in -
NUM_OF_BLOCKS*BLOCK_SIZE*DATA_SIZE;
    if (temp >= 0)
    {
      in_ptr = (short *)pcm_in;
    }

    temp = (int)out_ptr - pcm_out -
NUM_OF_BLOCKS*BLOCK_SIZE*DATA_SIZE;
    if (temp >= 0)
    {
      out_ptr = (short *)pcm_out;
    }
```

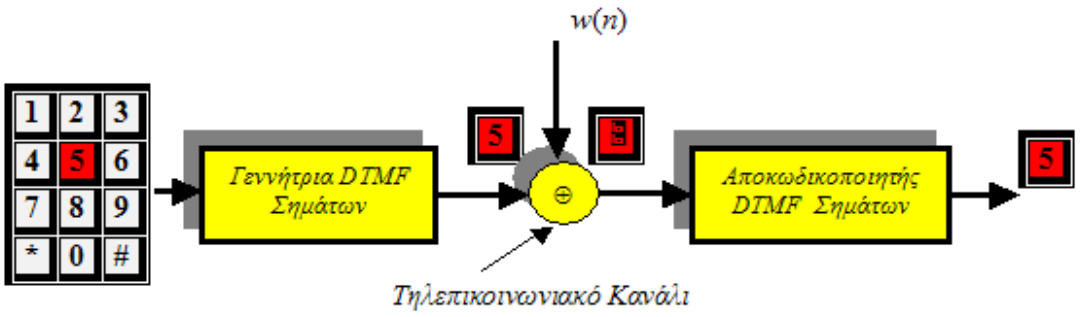
Ερώτημα 2: Με βάση το πρόγραμμα codec_edma.c που σας δίνεται καλείστε να δημιουργήσετε στο Code Composer Studio ένα πρόγραμμα αποκωδικοποίησης ενός DTMF σήματος που θα βασίζεται στον υπολογισμό του Διακριτού Μετασχηματισμού Fourier (DFT) του DTMF σήματος, χρησιμοποιώντας τον αλγόριθμο του Goertzel. Σχολιάστε τον αλγόριθμο Goertzel και περιγράψτε τον ακριβή τρόπο υλοποίησης που κάνατε.

Ερώτημα 3: Επαναλάβετε το προηγούμενο ερώτημα χρησιμοποιώντας για την αποκωδικοποίηση του DTMF σήματος την ιδέα της συστοιχίας φίλτρων (Filter Bank).

Ερώτημα 4: Υπολογίστε την πολυπλοκότητα των δυο τεχνικών αποκωδικοποίησης του DTMF σήματος που υλοποιήσατε στα δύο προηγούμενα ερωτήματα, και συμπεράνετε ποια είναι πιο οικονομική.

4. Συμπεριφορά του συστήματος παρουσία AWG θορύβου

Θεωρήστε τώρα ότι το τηλεπικοινωνιακό κανάλι δεν είναι ιδανικό αλλά είναι όπως παρουσιάζεται στο Σχήμα 4. Στην περίπτωση αυτή το σήμα που φθάνει στην είσοδο του αποκωδικοποιητή προκύπτει από την υπέρθεση του σήματος DTMF και του σήματος θορύβου w(n) ο οποίος θα θεωρήσουμε ότι είναι λευκός και Γκαουσιανός.



Σχήμα 4

Ερώτημα 5: Επαναλάβετε τη διαδικασία της αποκωδικοποίησης όπως στο Ερώτημα 2 και 3 έχοντας όμως προσθέσει στο σύστημα θόρυβο. Πειραματιστείτε για τιμές του SNR στο διάστημα [-25 0] με βήμα 2 dB, και δείτε ποια είναι η κατώτερη τιμή για την οποία τα συστήματά σας λειτουργούν ικανοποιητικά (σχετικά μικρό SER). Καταχωρήστε τα αποτελέσματά σας σε πίνακα.

Ερώτημα 6: Προσπαθήστε να αιτιολογήσετε θεωρητικά τα αποτελέσματά σας.

5. Βιβλιογραφία

[1] TMS320C6711 Digital Signal Processor Data Sheet (Literature Number SPRS088), Texas Instruments, Dallas, TX, 2002.

[2] TMS320C6000 Peripherals Guide (Literature Number SPRU190), Texas Instruments, Dallas, TX, 2001.

[3] TLC320AD535C/I Data Manual Dual Channel Voice/Data Codec, (Literature Number SLAS202), Texas Instruments, Dallas, TX, 1999.

[4] Code Composer Studio Getting Started Guide (Literature Number SPRU509), Texas Instruments, Dallas, TX, 2001.

[5] Γ. Β. Μουστακίδης, "Βασικές Τεχνικές Ψηφιακής Επεξεργασίας Σημάτων,", Εκδόσεις Τζιόλα 2004.

[6] Sanjit K. Mitra, "Digital Signal Processing, A Computer-Based Approach," McGraw-Hill, 1998.

[7] Vinay K. Ingle and John G. Proakis, "Digital Signal Processing Using MATLAB®," PWS Publishing Company, 1997.

[8] Steven A. Tretter, Communication System Design Using DSP Algorithms. With Laboratory Experiments for the TMS320C6701 and TMS320C6711, Kluwer Academic / Plenum Publishers, New York 2003.

[9] Brian L. Evans, “A Low-Cmplexity ITU-Compliant Dual Tone Multiple Frequency Detector”, IEEE Trnasaction on Signal Procesing, Oct. 1999,

[10] Matthew D. Felder, James C. Mason, and Brian L. Evans, “Efficient Dual-Tone Multifrequency Detection Using the Nonuniform Discrete Fourier Transform,” IEEE Signal Processing Letters, Vol. 5, No. 7, pp. 160-163, July 1998.