ΕΡΓΑΣΤΗΡΙΟ ΨΗΦΙΑΚΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΣΗΜΑΤΩΝ

Εργαστήριο 5

Ομάδα 09 – Group 1

ΕΠΩΝΥΜΟ	ΟΝΟΜΑ	ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ
ΔΑΣΟΥΛΑΣ	ΙΩΑΝΝΗΣ	1053711
ΔΟΥΡΔΟΥΝΑΣ	ΑΡΙΣΤΕΙΔΗΣ ΑΝΑΡΓΥΡΟΣ	1047398

Εισαγωγή στον προγραμματισμό χρησιμοποιώντας γλώσσα C στο περιβάλλον του CCS

Στόχος:

Η εισαγωγή στον προγραμματισμό του C6713 DSK board με γλώσσα C, χρησιμοποιώντας τον compiler της Texas Instruments. Δίνεται ένα πρόγραμμα που δεν είναι ολοκληρωμένο για να χρησιμοποιηθεί σαν βάση για να δουλέψετε και να ολοκληρώσετε την άσκηση. Το πρόγραμμα, με μία αρχική τροποποίηση, θα χρησιμοποιεί τη μέθοδο polling για να παίρνει τα δεδομένα από το codec ήχου (AIC23) και να τα βγάζει στο μεγάφωνο. Με μια δεύτερη τροποποίηση, θα δημιουργεί ηχώ στην φωνή σας, όταν μιλάτε στο μικρόφωνο.

Ασκήσεις:

Άσκηση 5.1

Να σχολιαστεί ο σκοπός της κάθε συνάρτησης

Συνάρτηση Bargraph:

Η συνάρτηση Bargraph χρησιμοποιείται για την αναπαράσταση σε ραβδόγραμμα της έντασης του ήχου σε βήματα των 6dB. Ένα short int μπορεί να λάβει τιμές από -32768 μέχρι +32767. Ωστόσο ο αριθμός -32768 δε μπορεί απλά να μετατραπεί σε + 32767 οπότε αντιμετωπίζεται σαν ξεχωριστή περίπτωση με το πρώτο if. Στη συνέχεια διακρίνονται περιπτώσεις για τις τιμές της εισόδου. Έτσι ανάλογα με το αν η είσοδος είναι μεγαλύτερη των MAXIMUM_INPUT/2, MAXIMUM_INPUT/4, MAXIMUM_INPUT/8 , τότε η τιμή της έντασης είναι -6 db,-12 db ή-18 db και ανάβουν 3 LEDS , 2 LEDS ή 1 LED αντίστοιχα. Σε περίπτωση που η τιμή εισόδου δεν καλύπτεται από της παραπάνω συγκρίσεις, δεν ανάβει κανένα Led

Συνάρτηση switches:

Η συνάρτηση αυτή διαβάζει και επιστρέφει την κατάσταση των διακοπών. Πιο συγκεκριμένα διαβάζει από το I/O port την τιμή των διακοπτών και στη συνέχεια αναθέτει στην μεταβλητή new_switch_value την τιμή των διακοπτών αφού πρώτα κάνει ένα shift right 4 και ένα AND με τον αριθμό 7 προκειμένου να λάβουν οι μεταβλητές τιμές από 0 έως 7.

Συνάρτηση stereo_to_mono:

Η συνάρτηση μετατρέπει δυο στερεοφωνικές εισόδους σε μία μονοφωνική. Αυτό επιτυγχάνεται μέσω της μεταβλητής temp η οποία αθροίζει τις δύο εισόδους και στη συνέχεια τις διαιρεί με το 2 έτσι ώστε έχει τιμή ίση με τον μέσο όρο των δύο σημάτων.

Αρχείο interrupts:

Το αρχείο αυτό περιέχει τις συναρτήσεις που κάνουν τις απαραίτητες αρχικοποιήσεις για τα interrupts. Η void timer0_interrupt_enable(void) ενεργοποιεί το INT 14 και το NMI για το timer 0 γράφοντας κατευθείαν στον καταχωρητή IER . Η void timer1_interrupt_enable(void) ενεργοποιεί το INT 15 και το NMI για το timer 1 γράφει στο IER . Η void global_interrupts_enable(void) ενεργοποιεί όλα τα interrupt.

Η void global_interrupts_disable(void) απενεργοποιεί όλα τα υπόλοιπα int και η individual_interrupts_disable(void) απενεργοποιεί τα individual interrupts. ΟΙ δύο αυτές συναρτήσεις γράφουν στους reg CSR και IER τις τιμές 0x0100 και 0x0001 αντίστοιχα. Τέλος η void pending_interrupts_clear(void) κάνει clear όλα τα interrupts που μένουν γράφοντας στον ICR την τιμή 0xFFFF.

Άσκηση 5.2 – Συμπλήρωση συναρτήσεων

Συμπληρώστε κατάλληλα την συνάρτηση main() στο αρχείο “DelaysandEcho.c”, έτσι ώστε το πρόγραμμα να χρησιμοποιεί την μέθοδο polling για να περνάει τα δεδομένα εισόδου στην έξοδο. Για τον AIC23 codec, χρησιμοποιούνται οι εξής δύο συναρτήσεις για την λήψη και την μετάδοση δεδομένων σημάτων:

• Int16 DSK6713_AIC23_read(DSK6713_AIC23_CodecHandle hCodec, Uint32 *val) : Η συνάρτηση αυτή χρησιμοποιείται για την λήψη των τιμών του σήματος εισόδου (ανάγνωση μίας τιμής του σήματος εισόδου, μήκους 32-bit, ανά στιγμή δειγματοληψίας). Το πρώτο όρισμα της συνάρτησης είναι ήδη αρχικοποιημένο στον κώδικα. Το δεύτερο όρισμα θα πρέπει να είναι η διεύθυνση που θα αποθηκεύετε σε κάθε στιγμή δειγματοληψίας την τιμή του σήματος εισόδου. Η συνάρτηση αυτή επιστρέφει μία τιμή flag που σχετίζεται με τον αν η σειριακή θύρα ολοκλήρωσε τη λήψη κάποιας τιμής του σήματος εισόδου και είναι έτοιμη να λάβει νέα.

• Int16 DSK6713_AIC23_write(DSK6713_AIC23_CodecHandle hCodec, Uint32 val) : Η συνάρτηση αυτή χρησιμοποιείται για την μετάδοση των τιμών του σήματος εξόδου (εγγραφή στον codec μίας τιμής του σήματος εξόδου, μήκους 32bit, ανά κλήση της συνάρτησης). Το πρώτο όρισμα της συνάρτησης είναι ήδη αρχικοποιημένο στον κώδικα. Το δεύτερο όρισμα θα πρέπει να είναι η τιμή του σήματος εξόδου κάθε χρονική στιγμή. Η συνάρτηση αυτή επιστρέφει μία τιμή flag που σχετίζεται με τον αν η σειριακή θύρα ολοκλήρωσε τη μετάδοση κάποιας τιμής του σήματος εξόδου και είναι έτοιμη να μεταδώσει νέα.

Για αυτό το ερώτημα, απλά αρχικοποιήθηκαν οι RCR, XCR καταχωρητές με την τιμή 00A0 για να ρυθμιστεί το κανάλι να λαμβάνει 32 bit σε μία φάση και να μεταδίδει 32 bit σε μία φάση, αντίστοιχα. Επίσης, ορίστηκε ένας 32-bit ακέραιος αριθμός val, ο οποίος χρησιμοποιείται για να διαβάζονται και να γράφονται τιμές συνεχώς μέσω των εντολών:
Int16DSK6713_AIC23_read(DSK6713_AIC23_CodecHandle hCodec, Uint32 *val), Int16 DSK6713_AIC23_write(DSK6713_AIC23_CodecHandle hCodec, Uint32 val) οι οποίες περιέχονται σε έναν ατέρμονα βρόχο.

Τα while loops των εντολών είναι έτσι ορισμνα, ώστε όσο δεν επιστρέφουν τιμή 0, που σημαίνει ότι επιτεύχθηκε η λειτουργία τους, να επαναλαμβάνεται η εντολή μέχρι να γίνει το read ή το write αντίστοιχα (γραμμές 87,88).

```
67  int main(void)
68  {
69
70      DSK6713_AIC23_CodecHandle hCodec;
71
72      // Initialize BSL
73      DSK6713_init();
74
75      //Start codec
76      hCodec = DSK6713_AIC23_openCodec(0, &config);
77
78      // Set  frequency to 48KHz
79      DSK6713_AIC23_setFreq(hCodec, DSK6713_AIC23_FREQ_48KHZ);
80
81      Uint32 val;
82
83      *(unsigned volatile int*)McBSP1_RCR = 0xA0;
84      *(unsigned volatile int*)McBSP1_XCR = 0xA0;
85
86      while(1){
87          while(DSK6713_AIC23_read(hCodec, &val) != 0);
88          while(DSK6713_AIC23_write(hCodec, val) != 0);
89      }
90
91      return (0);
92  }
93
```

Κατά την εκτέλεση, παρατηρήθηκε ότι το πρόγραμμα διάβαζε την τιμή του καταχωρητή A0 και την έγραφε έπειτα στον καταχωρητή A3, μέχρι ο χρήστης να τερματίσει την λειτουργία.

```
104
105     while(1){
106         while(!DSK6713_AIC23_read(hCodec, &val));
107         delay_array[current] = val;
108         next_echo = val & 0xFFFF0000;
109         prev_echo = delayed_input(delay_array, current);
110         val = prev_echo + next_echo;
111         current++;
112         if (current == N){
113             current = 0;
114         }
115
116         while(!DSK6713_AIC23_write(hCodec, val));
117     }
118     return (0);
119 }
```

Άσκηση 5.3

Να συμπληρωθεί η συνάρτηση delayed_input, που θα παίρνει ως είσοδο το καινούριο δείγμα εισόδου μαζί με την καθυστέρηση N και θα επιστρέφει το καθυστερημένο κατά N δείγμα. Η συνάρτηση θα χρησιμοποιεί κυκλικό buffering για λόγους ταχύτητας. Επίσης, να συμπληρωθεί η συνάρτηση delay_array_clear για την αρχικοποίηση του buffer. Χρησιμοποιώντας τον buffer καθυστέρησης, να σταλθεί ως έξοδος στο ένα κανάλι το τρέχον δείγμα εισόδου και στο άλλο κανάλι το κατάλληλο καθυστερημένο δείγμα. Στη συνέχεια, να πραγματοποιηθούν οι απαραίτητες αλλαγές, που περιγράφονται παραπάνω, για να λαμβάνεται ως είσοδος φωνή (από το μικρόφωνο). Να επαναληφθεί η διαδικασία, με σκοπό την δημιουργία ηχούς στην φωνή.

Οι προσθήκες και οι αλλαγές που έγιναν στον κώδικα του ερωτήματος 5.2 είναι αναφέρονται παρακάτω.

Αρχικά , πραγματοποιήθηκε αλλαγή στο DSK6713_AIC23_ANAPATH έτσι ώστε να ενεργοποιηθεί η είσοδος από το μικρόφωνο. Ο αριθμός που γράφτηκε είναι ο 0x15 και προκύπτει με βάση τα ακόλουθα.

```
28      0x0015,  /* 4 DSK6713_AIC23_ANAPATH    Analog audio path control */      \
```

BIT	D8	D7	D6	D5	D4	D3	D2	D1	D0
Function	STA2	STA1	STA0	STE	DAC	BYP	INSEL	MICM	MICB
Default	0	0	0	0	0	1	0	1	0

INSEL Input select for ADC 0 = Line 1 = Microphone

Αφού γίνει η αρχικοποίηση τον τιμών , συμπληρώνονται οι συναρτήσεις: delayed_input , delay_array_clear και main έτσι ώστε το πρόγραμμα να επιτελέσει την ζητούμενη λειτουργία.

Αρχικά , η delayed_input παίρνει σαν ορίσματα δύο μεταβλητές (int *array, int position) εκ των οποίων η πρώτη είναι ένας pointer που δείχνει το στοιχείο που έχουμε για είσοδο και η δεύτερη τη θέση που αποθηκεύεται η τελευταία είσοδος. Έπειτα ορίζεται μία νέα μεταβλητή με όνομα new που αυξάνει την τιμή του position κατά 1 και δίνει στην μεταβλητή echo την τιμή του στοιχείου που βρίσκεται στην επόμενη θέση. Όταν συνδέεται το μικρόφωνο, τα δεδομένα γράφονται σε έναν buffer. Έστω ότι η τιμή που λαμβάνεται σαν είσοδος είναι στην θέση position. Εάν η μεταβλητή echo γίνει ίση με το position τότε η συνάρτηση θα δώσει ως έξοδο το δεδομένο που γράφεται εκείνη την στιγμή στον buffer, με αποτέλεσμα να υπάρχει μηδενικό delay. Εάν όμως η echo λάβει την τιμή position +1 , δηλαδή εάν η συνάρτηση διαβάσει το επόμενο στοιχείο στον buffer , επειδή δεν έχει προλάβει να γραφτεί κάτι καινούργιο το στοιχείο που θα πάρει την τιμή echo θα είναι από τα προηγούμενα δεδομένα. Έχοντας πλέον μια 32 bit μεταβλητή τις καθυστερημένης εισόδου την οποία κάνουμε με shift right κατά 16 bit Το αποτέλεσμα που επιστρέφεται στην μεταβλητή echo είναι τα 16 MSB στις LSB θέσεις.

```
53  delayed_input(int *array, int position){
54      Uint32 echo;
55      int new;
56
57      new = position + 1;
58      echo = array[new];
59      echo = (short) (array[new] >> 16);
60      return echo;
61  }
```

Στη συνέχεια, συμπληρώνεται η συνάρτηση delay_array_clear(int *array) που ουσιαστικά μηδενίζει μέσω μία for loop τα στοιχεία του array που παίρνει σαν όρισμα.

```
68  delay_array_clear(int *array){
69      int i;
70      for(i=0; i<N; i++){
71          array[i] = 0;
72      }
73      return;
74  }
```

Τέλος, ορίζεται η συνάρτηση main στην οποία αφού αρχικοποιηθούν σωστά οι απαραίτητες μεταβλητές, προστίθεται ένας βρόχος while true σύμφωνα με τον οποίο μέχρι να διαβαστούν τα δεδομένα δίνεται την τιμή val στο στοιχείο του buffer που διαβάζεται. Στη συνέχεια η next_echo λαμβάνει τα 16 MSB της λέξης που διαβάστηκε. Έπειτα καλείται την συνάρτηση delayed_input που επιστρέφει την μεταβλητή echo η οποία προστίθεται στην val .Έτσι στο αριστερό κανάλι μεταδίδεται η γωνή χωρίς καθυστέρηση και στο δεξί η φωνή με καθυστέρηση. Τέλος με την χρήση μίας if γίνεται ο έλεγχος για να μην βγει το πρόγραμμα έξω από τον buffer. Μόλις διαβαστούν τα δεδομένα , η συνάρτηση DSK6713_AIC23_read(hCodec, &val) δίνει την τιμή 0 και το πρόγραμμα προχωράει στην επόμενη εντολή.

```
104
105     while(1){
106         while(!DSK6713_AIC23_read(hCodec, &val));
107         delay_array[current] = val;
108         next_echo = val & 0xFFFF0000;
109         prev_echo = delayed_input(delay_array, current);
110         val = prev_echo + next_echo;
111         current++;
112         if (current == N){
113             current = 0;
114         }
115
116         while(!DSK6713_AIC23_write(hCodec, val));
117     }
118     return (0);
119 }
```

Άσκηση 5.4

• Χρησιμοποιώντας την έτοιμη συνάρτηση user_switches_read για την ανάγνωση των dip switches, τροποποιείστε το πρόγραμμα, έτσι ώστε να επιλέγεται η καθυστέρηση σε πραγματικό χρόνο από τα switches. Η καθυστέρηση θα κυμαίνεται μεταξύ 0-4 δευτερολέπτων.

• Συμπληρώστε την συνάρτηση get_delay_time, για να παίρνετε την κατάλληλη καθυστέρηση ανάλογα με την κατάσταση των switches.

• Τέλος, συμπληρώστε και χρησιμοποιείστε την συνάρτηση switch_status_display, για να εμφανίζεται στο stdout η κατάσταση των switches, κάθε φορά που τους αλλάζουμε τιμή.

Συνάρτηση int main():

```c
int main(void)
{
    DSK6713_AIC23_CodecHandle hCodec;
    // Initialize BSL
    DSK6713_init();

    //Start codec
    hCodec = DSK6713_AIC23_openCodec(0, &config);

    // Set  frequency to 48KHz
    DSK6713_AIC23_setFreq(hCodec, DSK6713_AIC23_FREQ_48KHZ);

    int current = 0;
    Uint32 val, prev_echo, next_echo;
    short int prev_state = 0, next_state;
    float delay;

    delay_array_clear(delay_array);


    *(unsigned volatile int*)McBSP1_RCR = 0xA0;
    *(unsigned volatile int*)McBSP1_XCR = 0xA0;

    while(1){
        while(!DSK6713_AIC23_read(hCodec, &val));
        delay_array[current] = val;
        next_echo = val & 0xFFFF0000;
        next_state = user_switches_read();

        if((prev_state - next_state)!= 0){
            switch_status_display(next_state);
        }
        prev_state = next_state;
        delay = get_delay_time(next_state);

        prev_echo = delayed_input(delay_array, current, delay);
        val = prev_echo + next_echo;
        current++;
        if (current == N){
            current = 0;
        }
        while(!DSK6713_AIC23_write(hCodec, val));
    }
    return (0);
}
```

Αρχικά, γίνονται οι απαραίτητες αρχικοποιήσεις που θα χρησιμεύσουν στην συνέχεια και μηδενίζεται ο buffer με την συνάρτηση delay_array_clear(delay_array) όπως προηγουμένως.

Στο while loop δίνεται κάθε φορά η 32-bit τιμή val σε μία θέση του πίνακα και έπειτα με λογικό ‘and’ διατηρούνται τα 16MSBs τα οποία αποθηκεύονται στην μεταβλητή next_echo ώστε να μεταδίδονται από το αριστερό κανάλι. Στο αριστερό κανάλι δηλαδή θα γράφεται αυτούσια η πληροφορία που διαβάζεται και στο δεξί θα γραφτεί η ηχώ.

Ύστερα, στην μεταβλητή next_state αποθηκεύεται μία τιμή από 0 εώς 7 που δίνεται από την συνάρτηση user_switches_read() ανάλογα με την κατάσταση των διακοπτών του board. Με την if συνθήκη που ακολουθεί ελέγχεται αν υπήρχε μεταβολή στην κατάσταση των switches ελέγχοντας την next_state τιμή με την prev_state που είναι πάντα η προηγούμενη κατάσταση των διακοπτών. Η prev_state αρχικοποιείται ως prev_state = 0 ώστε να ανιχνευθεί η πρώτη μη μηδενική μεταβολή, ενώ στην συνέχεια λαμβάνει την τιμή next_state για να συγκριθεί ξανά στην επόμενη επανάληψη του βρόχου. Αν υπάρχει μεταβολή της κατάστασης, καλείται η συνάρτηση switch_status_display(next_state) όπου γίνεται μία εκτύπωση στο stdout της ενημερωμένης κατάστασης των διακοπτών.

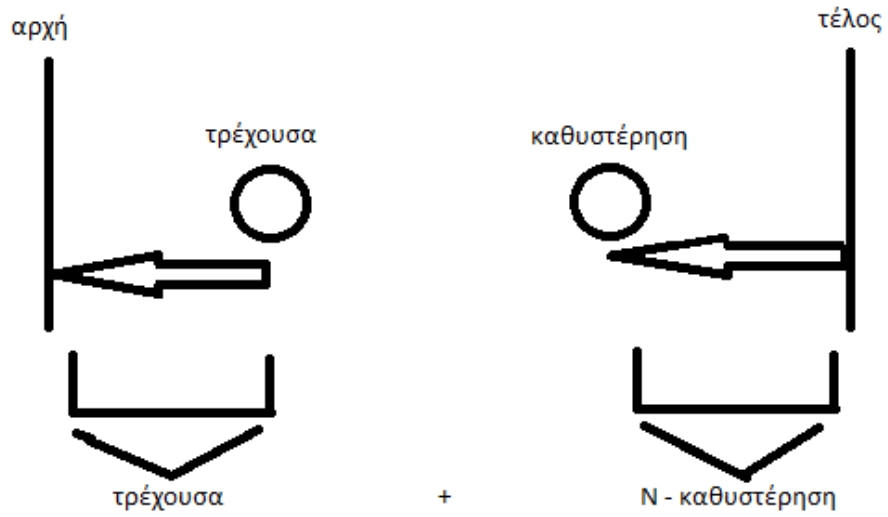
```c
switch_status_display(int state){
    printf("Current state is:%d\n", state);
    return;
}

```

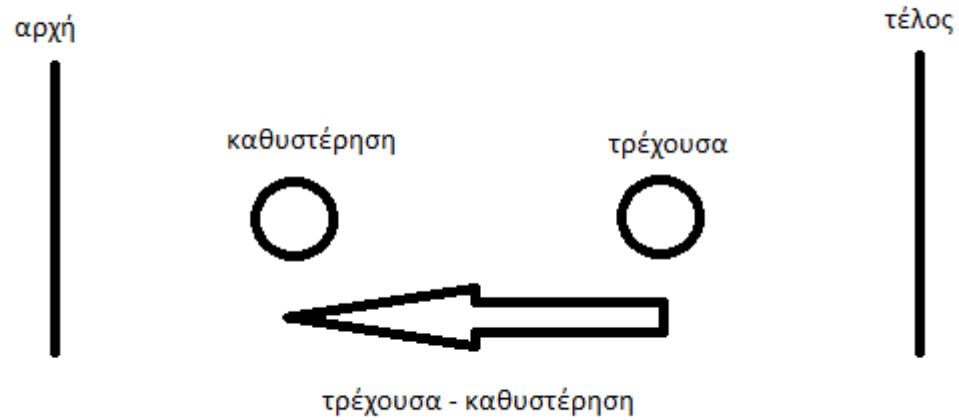
Έπειτα, δημιουργείται η ηχώ. Πρώτα πρέπει να βρεθεί η καθυστέρηση ανάλογα με την τιμή των διακοπτών. Αυτό γίνεται στην συνάρτηση get_delay_time(next_state). Ο delay_array αποτελείται από N στοιχεία. Για να δημιουργηθούν 8 διαφορετικές καθυστερήσεις, μία για κάθε κατάσταση διακοπτών (από 0 εώς 7), χωρίζεται το N array σε 8 delays με μία διαίρεση με το 7 (0*N, N/7, 2N/7, …, N). Επομένως, για κάθε κατάσταση διακοπτών δημιουργείται ένα διαφορετικό float delay που σώζεται στην μεταβλητή delay.

```c
get_delay_time(int state){
    float delay;
    delay = (state/7.0) * N;
    return delay;
}
```

Μετά, στην μεταβλητή prev_echo αποθηκεύεται η ηχώ μέσω της συνάρτησης delayed_input (delay_array, current, delay) έχοντας βρει πλέον το delay. Εκεί δίνεται μία τιμή στην μεταβλητή ho ανάλογα με την σχέση καθυστέρησης και τρέχουσας θέσης του πίνακα. Αρχικά, αν δεν υπάρχει καθυστέρηση, άρα και ηχώ, δίνεται στην ηχώ, η τρέχουσα θέση του πίνακα για να αναπαραχθεί δηλαδή η πληροφορία χωρίς καθυστέρηση. Αλλιώς, αν η καθυστέρηση είναι μεγαλύτερη από την τρέχουσα θέση του πίνακα, δίνεται η τιμή της θέσης: N – καθυστέρηση + τρέχουσα θέση + 1. Το + 1 προστίθεται για τον ίδιο λόγο με την άσκηση 5.3. Σχηματικά:



Αν, όμως, η τρέχουσα θέση είναι μεγαλύτερη ή ίση με την καθυστέρηση, η τιμή για την ηχώ δίνεται ως: τρέχουσα – καθυστέρηση + 1. Σχηματικά:



Έπειτα, η τιμή της νέας θέσης του πίνακα αποθηκεύεται στην τιμή echo και γίνεται ένα shft right 16 για να αποθηκευτούν τα 16 MSBs της στις LSB θέσεις ώστε να γραφτούν έπειτα στο δεξιό κανάλι.

```
55  delayed_input(int *array, int current, float delay){
56      Uint32 echo;
57      int new;
58
59      if (delay==0){
60          new = current;
61      }
62      else{
63          if(delay>current){
64              new = current - delay + N + 1;
65          }else{
66              new = current - delay + 1;
67          }
68      }
69
70      echo = array[new];
71      echo = (short) (echo >> 16);
72      return echo;
73  }
74
```

Πριν γίνει η εγγραφή, αποθηκεύεται στην τιμή val το άθροισμα prev_echo + next_echo, δηλαδή η val έχει στα 16 MSBs την τιμή της εισόδου και στα 16 LSBs την ηχό για αριστερό και δεξί κανάλι, αντίστοιχα. Πριν το τέλος του βρόχου, προστίθεται 1 στην τρέχουσα θέση του πίνακα για την επόμενη επανάληψη με την συνθήκη πως αν φτάσει η τρέχουσα θέση την τιμή N, να μηδενιστεί για να αρχίσει να διατρέχει τον buffer από την αρχή. Οι καταστάσεις των LED που προκύπτουν από τις καταστάσεις των διακοπών είναι συνολικά 8:

- Current State: 7 όταν όλα τα switches είναι επάνω
- Current State: 6 όταν το switch 0 είναι κάτω και τα 1 και 2 είναι επάνω
- Current State: 5 όταν το switch 1 είναι κάτω και τα 0 και 2 είναι επάνω
- Current State: 4 όταν το switch 2 είναι κάτω και τα 0 και 1 είναι επάνω
- Current State: 3 όταν τα switches 0 και 1 είναι κάτω και το 2 είναι επάνω
- Current State: 2 όταν τα switches 0 και 2 είναι κάτω και το 1 είναι επάνω
- Current State: 1 όταν τα switches 1 και 2 είναι κάτω και το 0 είναι επάνω
- Current State: 0 όταν όλα τα switches είναι επάνω

Βιβλιογραφία:

- **TMS320C67x/C67x+ DSP CPU and Instruction Set Reference Guide**
- **ΕΚΦΩΝΗΣΗ ΑΣΚΗΣΗΣ LAB-5 – ECLASS**
- **TMS320C6713 DATASHEET**
- **AIC23 Data Manual**