ΕΡΓΑΣΤΗΡΙΟ ΨΗΦΙΑΚΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΣΗΜΑΤΩΝ

## Εργαστήριο 5

## Εισαγωγή στον προγραμματισμό χρησιμοποιώντας γλώσσα C

## στο περιβάλλον του CCS

|  |
| --- |
| Ομάδα 17 7-12-2018 |
| Ασημακόπουλος Κωνσταντίνος 1046966 |
| Λουκαρέας Παύλος 1046970 |

**Περίληψη**

Σε αυτή την εργαστηριακή άσκηση εξοικειωθήκαμε με τον προγραμματισμό της πλακέτας σε γλώσσα C χρησιμοποιώντας τον compiler της ΤΙ. Αρχικά μας δόθηκε ένα ημιτελές πρόγραμμα πάνω στο οποίο έπρεπε να δουλέψουμε για να ολοκληρώσουμε την άσκηση. Ο σκοπός της άσκησης είναι να χρησιμοποιήσουμε τη μέθοδο polling για να παίρνουμε δεδομένα από το codec ήχου και να τα βγάζουμε στο μεγάφωνο καθώς και να δημιουργούμε ηχώ στη φωνή μας όταν μιλάμε στο μικρόφωνο.

**Άσκηση 5.1**

Σε αυτό το ερώτημα θα πρέπει να σχολιάσουμε την κάθε συνάρτηση του έτοιμου προγράμματος. Η πρώτη συνάρτηση που θα σχολιάσουμε είναι:

void delay\_array\_clear(int \*BUFFER){

int i;

for (i = 0; i < N; i++){

BUFFER[i] = 0;

}

return;

}

Η συνάρτηση αυτή αρχικοποιεί τον κυκλικό buffer με την τιμή 0 για τα συνολικά δείγματα στα 4 δευτερόλεπτα καθυστέρησης.

Παρακάτω έχουμε τις εντολές μέσα στην main συνάρτηση οι οποίες διαβάζουν την κατάσταση της ΙΟ θύρας και μας δίνει την καθυστέρηση με μια τιμή από 0-7. Η εντολή αυτή είναι η

status = user\_switches\_read();

Η επόμενη συνάρτηση που θα αναλύσουμε είναι η get\_delay\_time() η οποία παίρνει την τιμή 0-7 και την μεταφράζει σε καθυστέρηση 0-4 δευτερόλεπτα. Αυτό γίνεται με τον παρακάτω τρόπο:

float get\_delay\_time(int delay) {

    float time;

    time = delay\*4.0f/7.0f;

    return(time);

}

**Άσκηση 5.2**

Σε αυτό το ερώτημα συμπληρώσαμε κατάλληλα τη συνάρτηση main() έτσι ώστε το πρόγραμμα να χρησιμοποιεί τη μέθοδο polling για να περνάει τα δεδομένα εισόδου. Χρησιμοποιώντας τις δύο παρακάτω συναρτήσεις:

* DSK6713\_AIC23\_read(DSK6713\_AIC23\_CodecHandle hCodec, Uint32 \*val)

Η παραπάνω συνάρτηση χρησιμοποιείται για τη λήψη των τιμών του σήματος εισόδου (32bit). Το πρώτο όρισμα είναι ήδη αρχικοποιημένο ενώ το δεύτερο είναι η διεύθυνση που θα αποθηκεύεται σε κάθε στιγμή δειγματοληψίας την τιμή του σήματος εισόδου. Η συνάρτηση επιστρέφει μια τιμή που μας δείχνει αν η σειριακή θύρα είναι έτοιμη να λάβει νέα τιμή.

* DSK6713\_AIC23\_write(DSK6713\_AIC23\_CodecHandle hCodec, Uint32 val)

Η παραπάνω συνάρτηση χρησιμοποιείται για τη εγγραφή των τιμών εξόδου (32bit). Το πρώτο όρισμα είναι ήδη αρχικοποιημένο ενώ το δεύτερο είναι η τιμή του σήματος εξόδου σε κάθε στιγμή δειγματοληψίας. Η συνάρτηση επιστρέφει μια τιμή που μας δείχνει αν η σειριακή θύρα είναι έτοιμη να γράψει νέα τιμή.

Ο κώδικας που εκτελεί τα παραπάνω είναι ο εξής:

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);

\* (unsigned volatile int\*)McBSP1\_RCR = 0x000000A0;

\* (unsigned volatile int\*)McBSP1\_XCR = 0x000000A0;

while(1) {

while(!DSK6713\_AIC23\_read(hCodec, &val));

while(!DSK6713\_AIC23\_write(hCodec, val));

}

return (0);

}

**Άσκηση 5.3**

Σε αυτή την άσκηση όπως και στην επόμενη θέλουμε να δημιουργήσουμε ηχώ στη φωνή μας. Για το λόγο αυτό χρησιμοποιούμε τη συνάρτηση delayed\_input() η οποία παίρνει ως είσοδο το καινούργιο δείγμα εισόδου μαζί με την καθυστέρηση και επιστρέφει το καθυστερημένο δείγμα. Η συνάρτηση αυτή χρησιμοποιεί κυκλικό buffering για λόγους ταχύτητας. Η συνάρτηση είναι η παρακάτω:

unsigned int delayed\_input(float time, int position, int \*buffer) {

unsigned int echo;

int displacement, pos;

displacement = time \* 48000;

if (displacement == 0) {

    echo = buffer[position];

} else {

    if (displacement > position) {

        pos = N - displacement + position;

        echo = buffer[pos];

    } else {

        pos = position - displacement;

        echo = buffer[pos];

    }

}

echo = (short)(echo >> 16);

return echo;

}

Επίσης χρησιμοποιούμε τη συνάρτηση delay\_array\_clear() της οποίας ο ορισμός και η λειτουργία εξηγήθηκαν στην άσκηση 1.

**Άσκηση 5.4**

Όπως εξηγήθηκε και στο ερώτημα 1 χρησιμοποιούμε την έτοιμη συνάρτηση user\_switches\_read() για την ανάγνωση των DIP Switches η οποία μας επιστρέφει μια τιμή από 0 έως 7. Χρησιμοποιώντας τη συνάρτηση get\_delay\_time() μεταφράζουμε αυτή την καθυστέρηση σε χρόνο από 0 έως 4 δευτερόλεπτα.

float get\_delay\_time(int delay) {

    float time;

    time = delay\*4.0f/7.0f;

    return(time);}

Τέλος χρησιμοποιούμε την συνάρτηση switch\_status\_display() για να εμφανίσουμε στο stdout την κατάσταση των switches κάθε φορά που η τιμή τους αλλάζει.

void switch\_status\_display(int status) {

    printf("Present state is: %d\n", status);

}

Ο ολόκληρος κώδικας του προγράμματος επισυνάπτεται μαζί με την αναφορά.