Εργαστηριακή Άσκηση 3

Βοηθητική Παρουσίαση

Κώδικας - Επεξήγηση σε σχέση με το hardware - Μετατροπή για Εξομοίωση σε οποιοδήποτε IDE.

Δ.Π.Μ.Σ. Ηλεκτρονική και Επεξεργασία της Πληροφορίας

Πόθος Βασίλειος

```
#include <AT898252.H>
#include <stdio.h>
#define DAC_BUFFER_SIZE 500  //Buffer Size for DAC data.

char xdata DACdw _at_ 0xC000; //Variable refered to DAC memory address.
struct DACBuffer_struct {     //DAC struct. Buffer and Index of Buffer grouped.
        unsigned char Buf[DAC_BUFFER_SIZE];
        int Ind;
};
xdata struct DACBuffer_struct DAC;   //Declaration of DAC at xdata memory.

void intrpt(void) interrupt 1 using 3 { //Interrupt 1, using 3 bank.
        //While Index of Buffer for DAC < buffersize, increase it and sent data to DAC.
        //Else zero it.
        if (DAC.Ind + 1 > DAC_BUFFER_SIZE) { DAC.Ind = 0; }
        DACdw = DAC.Buf[DAC.Ind++];
}

void MakeSound(void) {    //Load DACBuffer with data..
        int i;
        for (i = 0; i < DAC_BUFFER_SIZE; i++) {
          (i < DAC_BUFFER_SIZE/2) ? (DAC.Buf[i] = i) : (DAC.Buf[i] = DAC_BUFFER_SIZE - i);
        }
}

void main (void) {
        char UserInput, TransFreq = 0xFF;

        SCON = 0x50;  //8-bit UART
        TMOD |= 0x22; //timer 1 at 8-bit auto-reload
        TH1 = 0xf4;   //Baud rate at 2400b/s (11.0592MHz clock)
        TR1 = 1;      //Start timer 1
        TI = 1;       //Ready for transmission
        IE = 0x82;    //Enable timer 0 interrupt
        TH0 = 0xFF;   //Default transmission frequency at 0xFF.

        MakeSound();  //Load DACBuffer with data..

        while(1) {
                //Ask Selection from user.
                printf("\n\n***Select Action:\n1) Set transmission frequency\n
                        2)Start.\n3) Stop\n4) Reset.\n");

                  scanf("%c",&UserInput);
                  if (UserInput == '1') {  //If pressed '1', Set TH0 to user preference.
                        printf(") Set TH0 value to: "); scanf("%d",&TransFreq); TH0=TransFreq;
                  } else if (UserInput == '2') {  //If pressed '2', Enable DAC interrupt. (Start)
                        TR0=1;
                  } else if (UserInput == '3') {  //If pressed '3', Disable DAC interrupt. (Stop)
                        TR0=0;
                  } else if (UserInput == '4') {  //If pressed '4', Reset Index of DAC Buffer.
                        DAC.Ind = 0;
                  }
        }
}
```

```
#include <AT898252.H>
#include <stdio.h>
#define DAC_BUFFER_SIZE 500  //Buffer Size for DAC data.
```

Ορίζεται το DAC_FUFFER_SIZE να είναι 500.

```
char xdata DACdw _at_ 0xC000; //Variable refered to DAC memory address.
struct DACBuffer_struct {     //DAC struct. Buffer and Index of Buffer grouped.
        unsigned char Buf[DAC_BUFFER_SIZE];
        int Ind;
};
xdata struct DACBuffer_struct DAC;   //Declaration of DAC at xdata memory.
```

α) Ορίζεται η μεταβλητή DACdw να βρίσκεται στην θέση μνήμης 0xC000. (Άλλη σελίδα μνήμης).

β) Στην συνέχεια φτιάχνω μια δομή η οποία περιέχει έναν πίνακα με 500 θέσης μνήμης, και μια μεταβλητή στην οποία θα αποθηκεύω κάθε φορά το νούμερο του κελιού από το οποίο θα διαβάζω τα δεδομένα.

γ) και τέλος φτιάχνω μια μεταβλητή του τύπου της δομής. Η μεταβλητή είναι η DAC.

```
void intrpt(void) interrupt 1 using 3 { //Interrupt 1, using 3 bank.
        //While Index of Buffer for DAC < buffersize, increase it and sent data to DAC.
        //Else zero it.
        if (DAC.Ind + 1 > DAC_BUFFER_SIZE) { DAC.Ind = 0; }
        DACdw = DAC.Buf[DAC.Ind++];
}
```

α) Κάθε φορά που καλείται αυτή η συνάρτηση ελέγχω αν το επόμενο κελί για διάβασμα δεδομένων είναι υπαρκτό ή όχι. Αν η τιμή του DAC.Ind +1 είναι μεγαλύτερη του 500, τότε το επόμενο κελί για διάβασμα δεδομένων θα είναι το 0.

β) Στην συνέχεια στέλνω τα δεδομένα του επόμενου κελιού στον DAC.

```
void MakeSound(void) {    //Load DACBuffer with data..
        int i;
        for (i = 0; i < DAC_BUFFER_SIZE; i++) {
          (i < DAC_BUFFER_SIZE/2) ? (DAC.Buf[i] = i) : (DAC.Buf[i] = DAC_BUFFER_SIZE - i);
        }
}
```

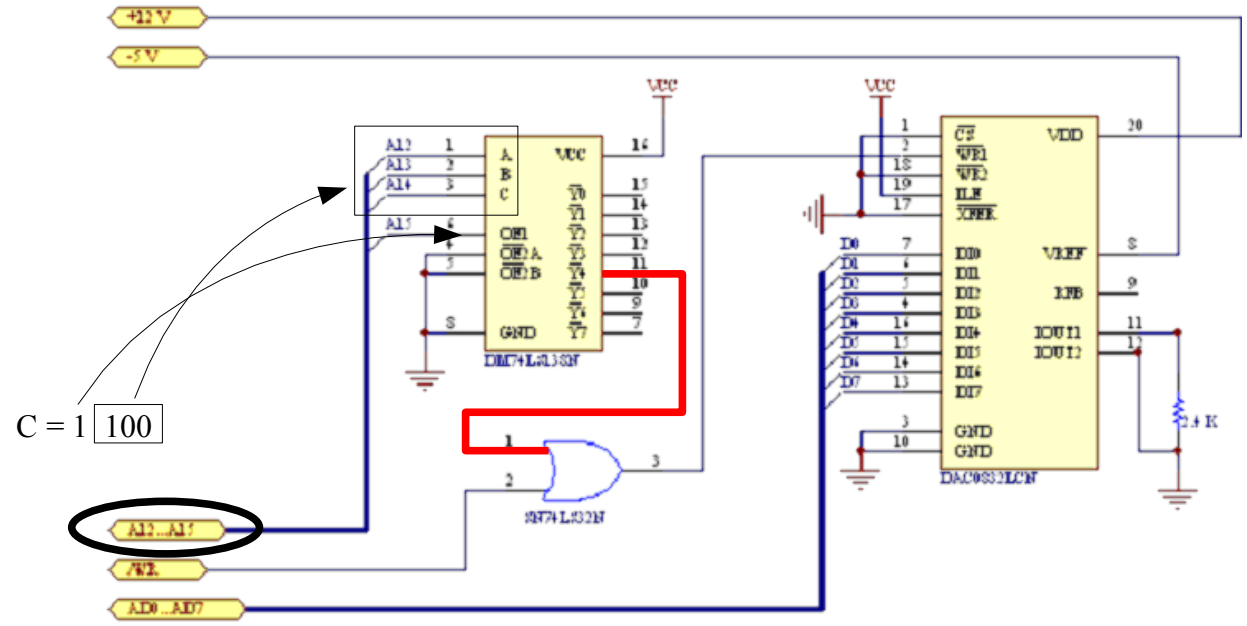
Γεμίζω τον πίνακα με τα δεδομένα μου (DAC.Buf[]) με δεδομένα τριγωνικής μορφής.

```
void main (void) {
        char UserInput, TransFreq = 0xFF;

        SCON = 0x50;  //8-bit UART
        TMOD |= 0x22; //timer 1 at 8-bit auto-reload
        TH1 = 0xf4;   //Baud rate at 2400b/s (11.0592MHz clock)
        TR1 = 1;      //Start timer 1
        TI = 1;       //Ready for transmission
        IE = 0x82;    //Enable timer 0 interrupt
        TH0 = 0xFF;   //Default transmission frequency at 0xFF.

        MakeSound();  //Load DACBuffer with data..

        while(1) {
                //Ask Selection from user.
                printf("\n\n***Select Action:\n1) Set transmission frequency\n
                        2)Start.\n3) Stop\n4) Reset.\n");

                scanf("%c",&UserInput);
                if (UserInput == '1') {  //If pressed '1', Set TH0 to user preference.
                        printf(") Set TH0 value to: "); scanf("%d",&TransFreq); TH0=TransFreq;
                } else if (UserInput == '2') {  //If pressed '2', Enable DAC interrupt. (Start)
                        TR0=1;
                } else if (UserInput == '3') {  //If pressed '3', Disable DAC interrupt. (Stop)
                        TR0=0;
                } else if (UserInput == '4') {  //If pressed '4', Reset Index of DAC Buffer.
                        DAC.Ind = 0;
                }
        }
}
```

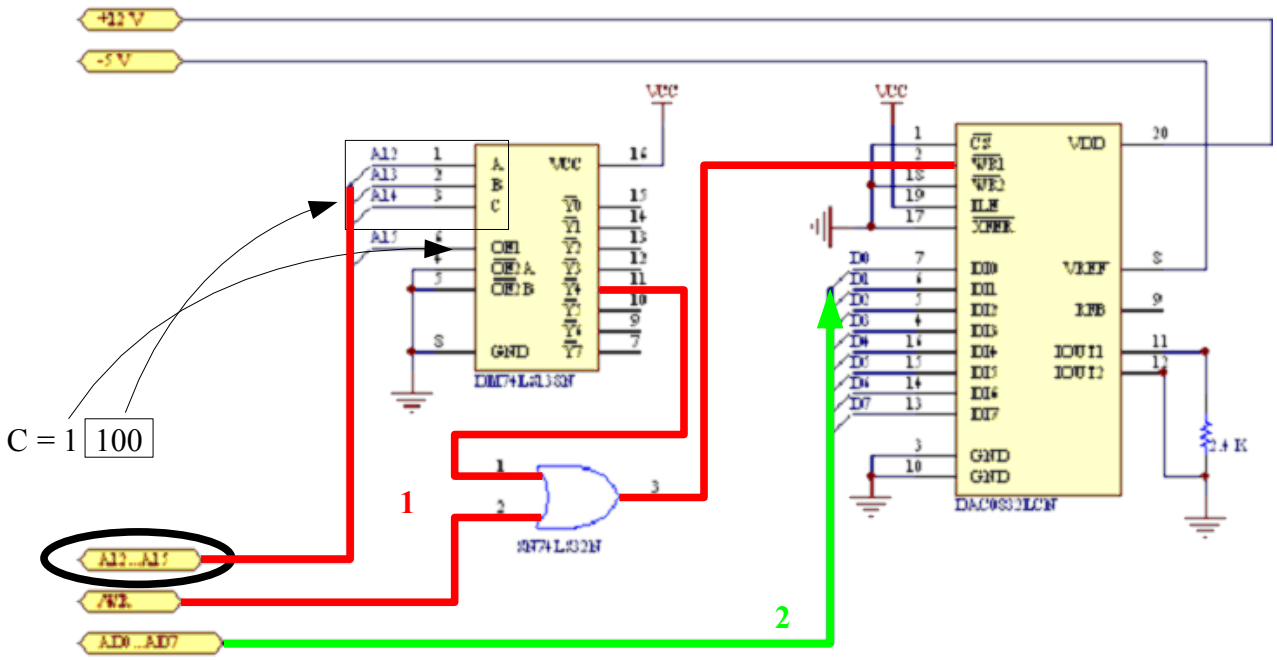
- Ορίζω 2 μεταβλητές. Την UserInput και την TransFreq. Στην πρώτη, αποθηκεύω της επιλογές του χρήστη. Στην δεύτερη αποθηκεύω την κατάλληλη τιμή για τον TH0 η οποία θα καθορίσει την συχνότητα κατα την οποία στέλνονται έξω τα δεδομένα.
- Καλώ την MakeSound() για να αρχικοποιήσω τον πίνακα δεδομένων μου.
- Ζητώ από τον χρήστη να επιλέξει μια από 4 επιλογές.
  - α) Συχνότητα αποστολής. Θέτω τον καταχωρητή TH0 με την τιμή του χρήστη. 0xFF → Μέγιστη συχνότητα.
  - β) Ενεργοποιώ τον DAC. Αυτό το κάνω ενεργοποιώντας τον Timer 0.
  - γ) Απενεργοποιώ τον DAC. Αυτό το κάνω απενεργοποιώντας τον Timer 0.
  - δ) Μηδενίζω τον δείκτη στα δεδομένα μου, για το επόμενο δεδομένο προς αποστολή. Δηλαδή ξεκινώ να ξαναστέλνω το σήμα μου στον DAC από την αρχή.

```
char xdata DACdw _at_ 0xC000; //Variable refered to DAC memory address.
```

Ο DAC ενεργοποιείται με αναφορά στην θέση 0xC000 όπως φαίνεται στο παρακάτω σχήμα.




```
void intrpt(void) interrupt 1 using 3 { //Interrupt 1, using 3 bank.
        //While Index of Buffer for DAC < buffersize, increase it and sent data to DAC.
        //Else zero it.
        if (DAC.Ind + 1 > DAC_BUFFER_SIZE) { DAC.Ind = 0; }
        DACdw = DAC.Buf[DAC.Ind++];
}
```



Διαδικασία Simulation

Στην διαδικασία αυτή προσπαθώ να εξομοιώσω τον κώδικα, ώστε να ελέγξω την σωστή λειτουργία του, χωρίς την ύπαρξη των επιμέρους υλικών μερών που απαιτούνται (ADC κτλ).

Μετατρέπω τον κώδικα από C51 σε ANSI C.

Η διαδικασία μπορεί να γίνει μέσω οποιουδήποτε IDE ανάπτυξης εφαρμογών C, με τις κατάλληλες αλλαγές.

```
#include <AT898252.H>
#include <stdio.h>
#define DAC_BUFFER_SIZE 500  //Buffer Size for DAC data.
```

Δεν χρησιμοποιώ το .H file το οποίο σχετίζεται με τον συγκεκριμένο microcontroller - αφού δεν τον έχω πρακτικώς.

```
char xdata DACdw; // _at_ 0xC000; //Variable refered to DAC memory address.
struct DACBuffer_struct {     //DAC struct. Buffer and Index of Buffer grouped.
        unsigned char Buf[DAC_BUFFER_SIZE];
        int Ind;
};
xdata struct DACBuffer_struct DAC;   //Declaration of DAC at xdata memory.
```

α) Δεν χρησιμοποιώ της λέξεις κλειδιά xdata καθώς στην εξομοίωση οι μεταβλητές θα αποθηκευτούν σε θέσης μνήμη υπολογιστή.

β) Με την ίδια λογική δεν χρησιμοποιώ την λέξη κλειδί "_at_".

```
void intrpt(void) { // interrupt 1 using 3 { //Interrupt 1, using 3 bank.
        //While Index of Buffer for DAC < buffersize, increase it and sent data to DAC.
        //Else zero it.
        if (DAC.Ind + 1 > DAC_BUFFER_SIZE) { DAC.Ind = 0; }
        DACdw = DAC.Buf[DAC.Ind++];
}
```

Αφαιρούμε τις λέξεις κλειδιά της C51. Την συνάρτηση αυτή θα την καλούμε εμείς μέσα από το πρόγραμμα, εξομοιώνοντας την λειτουργία της σαν να την καλούσε διακοπή.

```
void MakeSound(void) {    //Load DACBuffer with data..
        int i;
        for (i = 0; i < DAC_BUFFER_SIZE; i++) {
          (i < DAC_BUFFER_SIZE/2) ? (DAC.Buf[i] = i) : (DAC.Buf[i] = DAC_BUFFER_SIZE - i);
        }
}
```

Η Συνάρτηση είναι σε ANSI C. Δεν χρειάζεται αλλαγή.

```
void main (void) {
        char UserInput, TransFreq = 0xFF;
        char Flag_inter = 1;

        SCON = 0x50;  //8-bit UART
        TMOD |= 0x22; //timer 1 at 8-bit auto-reload
        TH1 = 0xf4;   //Baud rate at 2400b/s (11.0592MHz clock)
        TR1 = 1;      //Start timer 1
        TI = 1;       //Ready for transmission
        IE = 0x82;    //Enable timer 0 interrupt
        TH0 = 0xFF;   //Default transmission frequency at 0xFF.

        MakeSound();  //Load DACBuffer with data..

        while(1) {
                //Ask Selection from user.
                printf("\n\n***Select Action:\n1) Set transmission frequency\n
                        2)Start.\n3) Stop\n4) Reset.\n");
                while (Flag_inter) {
                        intrpt();
                }
                scanf("%c",&UserInput);
                if (UserInput == '1') {  //If pressed '1', Set TH0 to user preference.
                        printf(") Set TH0 value to: "); scanf("%d",&TransFreq); TH0=TransFreq;
                } else if (UserInput == '2') {  //If pressed '2', Enable DAC interrupt. (Start)
                        TR0=1; Flag_inter=1;
                } else if (UserInput == '3') {  //If pressed '3', Disable DAC interrupt. (Stop)
                        TR0=0; Flag_inter=0;
                } else if (UserInput == '4') {  //If pressed '4', Reset Index of DAC Buffer.
```

```
                    DAC.Ind = 0;
                }
        }
}
```

Οτιδήποτε εντολές αναφέρονται σε Hardware δεν χρησιμοποιούνται. Άλλωστε υποτίθεται αυτές οι εντολές δουλεύουν σωστά &ξέρουμε την λειτουργία τους, επομένως δεν χρειάζεται να τις εξομοιώσουμε.

Στην συνέχεια προσθέτουμε τις εξής μεταβλητές:

Char_Flag = 1; → Είναι αντίστοιχη με την TR0. Αν είναι 1, έχουμε interrupts. Αν είναι 0 τότε δεν έχουμε.

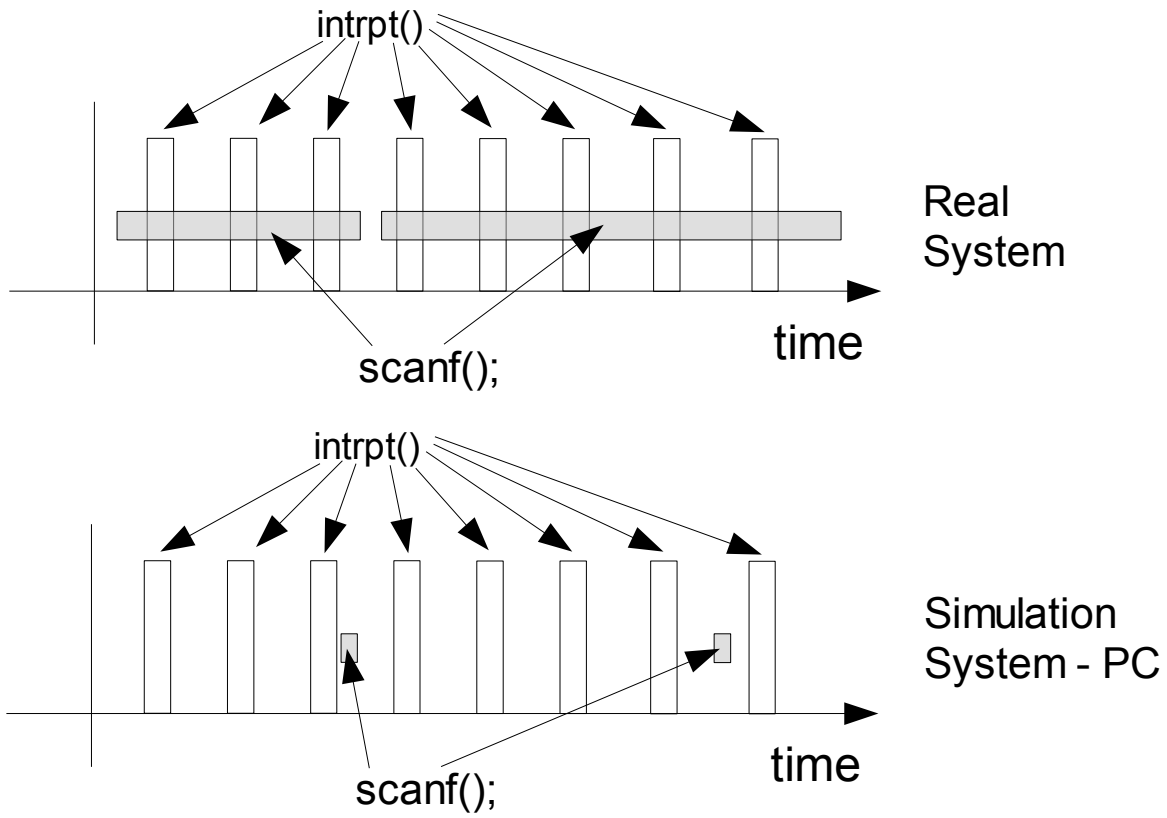
while (Flag_inter) { intrpt(); } → Πρέπει να μπει προτού την scanf. Εδώ υπάρχει ένα πρόβλημα.

Στο πραγματικό κύκλωμα θα τρέχει η συνάρτηση scanf() κάνοντας διαδικασία Polling μέχρι να πατηθεί το πλήκτρο enter και να παρθούν τα δεδομένα. Αμέσως μετά θα ξαναεκτελείται από την επανάληψη του while. Η διαδικασία αυτή θα σταματάει από interrupt στο σύστημα, προκειμένου να καλεστεί η intrpt().

Στην εξομείωση μέσω της ANSI C, δεν έχουμε την δυνατότητα να οδηγήσουμε την ροή του προγράμματος αλλού, όπως θα έκανε μια interrupt. Αν η ροή του προγράμματος φτάσει στην scanf() μετά για να βγει από εκεί, θα πρέπει να πατήσει ο χρήστης enter. Επομένως εδώ ακολουθείται "ανάποδη" λογική που όμως δεν αλλάζει το αποτέλεσμα του κώδικα.

Με αυτή την λογική, το πρόγραμμα καλεί συνεχόμενα την intrpt() σαν να την καλούσε μια διακοπή. Όταν ο χρήστης αποφασίζει πως θέλει να πληκτρολογίσει κάτι, τότε μέσω τις εξομοίωσης αλλάζουμε την μεταβλητή Flag_Inter σε 0, κάνοντας έτσι μια νοητή interrupt για να δώσει ο χρήστης δεδομένα. Αμέσως μετά την εισαγωγή δεδομένων επαναφέρουμε την Flag_Inter στην προηγούμενη κατάσταση, και συνεχίζει η εκτέλεση του προγράμματος.

Ουσιάστικα, είναι σαν να συμπιέζουμε όλο τον χρόνο της εισαγωγής δεδομένων από τον χρήστη σε μια στιγμή ανάμεσα σε 2 διαδοχικά interrupts για την εξαγωγή δεδομένων. Είναι διαφορετική διαδικασία απ αυτή που γίνεται στην πραγματικότητα, αλλά είναι απολύτως συμβατή για την λογική αποσφαλμάτωση του συγκεκριμένου κώδικα.



Τελικός κώδικας για εξομοίωση:

```
#include <stdio.h>
#define DAC_BUFFER_SIZE 500  //Buffer Size for DAC data.
char DACdw;
struct DACBuffer_struct {     //DAC struct. Buffer and Index of Buffer grouped.
        unsigned char Buf[DAC_BUFFER_SIZE];
        int Ind;
};
struct DACBuffer_struct DAC;   //Declaration of DAC at xdata memory.
void intrpt(void) { // interrupt 1 using 3 { //Interrupt 1, using 3 bank.
        //While Index of Buffer for DAC < buffersize, increase it and sent data to DAC.
        //Else zero it.
        if (DAC.Ind + 1 > DAC_BUFFER_SIZE) { DAC.Ind = 0; }
        DACdw = DAC.Buf[DAC.Ind++];
}
void MakeSound(void) {    //Load DACBuffer with data..
        int i;
        for (i = 0; i < DAC_BUFFER_SIZE; i++) {
          (i < DAC_BUFFER_SIZE/2) ? (DAC.Buf[i] = i) : (DAC.Buf[i] = DAC_BUFFER_SIZE - i);
        }
}
void main (void) {
        char UserInput, TransFreq = 0xFF;
        char Flag_inter = 1;


        MakeSound();  //Load DACBuffer with data..

        while(1) {
                //Ask Selection from user.
                printf("\n\n***Select Action:\n1) Set transmission frequency\n
                        2)Start.\n3) Stop\n4) Reset.\n");
                while (Flag_inter) {
                        intrpt();
                }
                scanf("%c",&UserInput);
                if (UserInput == '1') {  //If pressed '1', Set TH0 to user preference.
                        printf(") Set TH0 value to: "); scanf("%d",&TransFreq);
                } else if (UserInput == '2') {  //If pressed '2', Enable DAC interrupt. (Start)
                        Flag_inter=1;
                } else if (UserInput == '3') {  //If pressed '3', Disable DAC interrupt. (Stop)
                        Flag_inter=0;
                } else if (UserInput == '4') {  //If pressed '4', Reset Index of DAC Buffer.
                        DAC.Ind = 0;
                }
        }
}
```