Εργαστηριακή Άσκηση 2

Βοηθητική Παρουσίαση

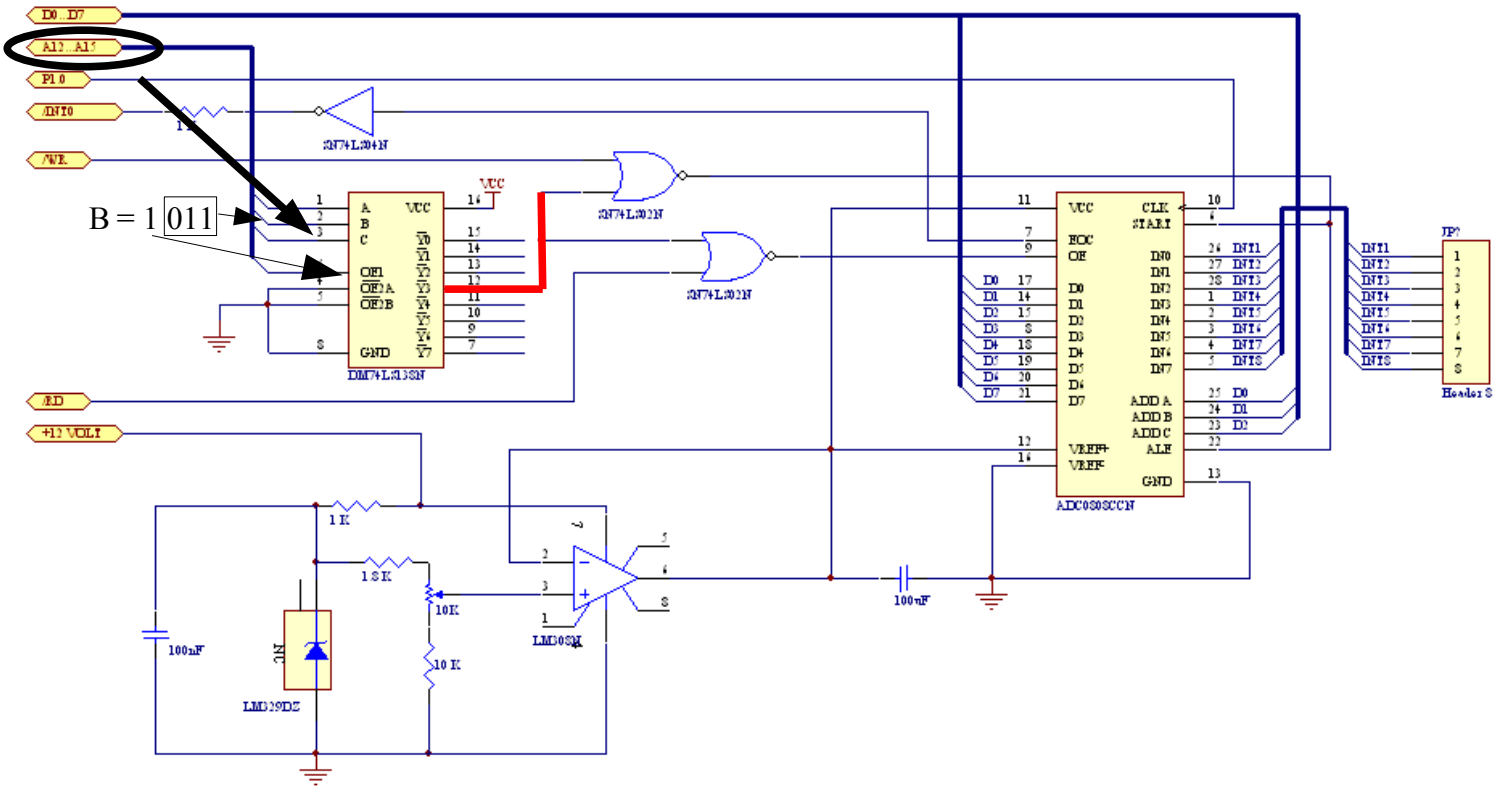
Κώδικας - Επεξήγηση σε σχέση με το hardware - Μετατροπή για Εξομοίωση σε οποιοδήποτε IDE.

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

Πόθος Βασίλειος

```
#include <AT898252.H>
#include <stdio.h>
#include <ctype.h>

unsigned char ADCrd _at_ 0xB000;

struct record {
	unsigned char input;
	unsigned char value;
};

xdata struct record ADC;
bit ADC_Busy = 1;

int uchar2int(unsigned char x) {
	int i; i=(int)x-48;
	return i;
}

void ReadADCData(void) interrupt 0 using 1 {
	ADC.value = ADCrd;
	ADC_Busy = 0;
}

void SetADCInput(unsigned char in) {
	ADC.input = in;
	ADCrd = in;
	ADC_Busy = 1;
}

void InitADC(void) {
	C_T2=0;				//Init Clock from T2
	T2MOD=2;			//Init Clock from T2
	RCAP2H=0xFF;			//500KHz
	RCAP2L=0xFA;			//500KHz
	TR2=1;				 //Timer 2 Start
	//----------------------
	IT0=1; //Edge trigered interruption on (INT0)
	EX0 = 1; // Set on /INT0 for ADC
	//----------------------
}

void main(void) {
unsigned int x=0;
	SCON = 0x50;
	TMOD |= 0x20;
	TH1 = 0xf4;	//Set BaudRate at 2400b/s (11.0592MHz clock)
	TR1 = 1;
	TI = 1;
	InitADC();	//Initialization of ADC
	EA=1;		//All interupts enabled.

	while(1) {
		printf ("Insert the number of the analog input\n");
		while((x=uchar2int(getchar()))>8 || x<0) {
			printf ("\n the number must be between 1 and 8 \n");
		}
		SetADCInput(x);
		while(ADC_Busy);
		printf ("\n AN %d: %d\n",(unsigned int)ADC.input,(unsigned int)ADC.value);
	}
}
```

```
unsigned char ADCrd _at_ 0xB000;
```




```
struct record {
        unsigned char input;
        unsigned char value;
};

xdata struct record ADC;
bit ADC_Busy = 1;
```

→ Ορίζω δομή: 2 μεταβλητές υπό ένα όνομα (Καλύτερη διαχείρηση)

→ Η δομή αυτή είναι η ADC στην xdata μνήμη

Ορίζω και μία μεταβλητή τύπου bit.

Οι συναρτήσεις

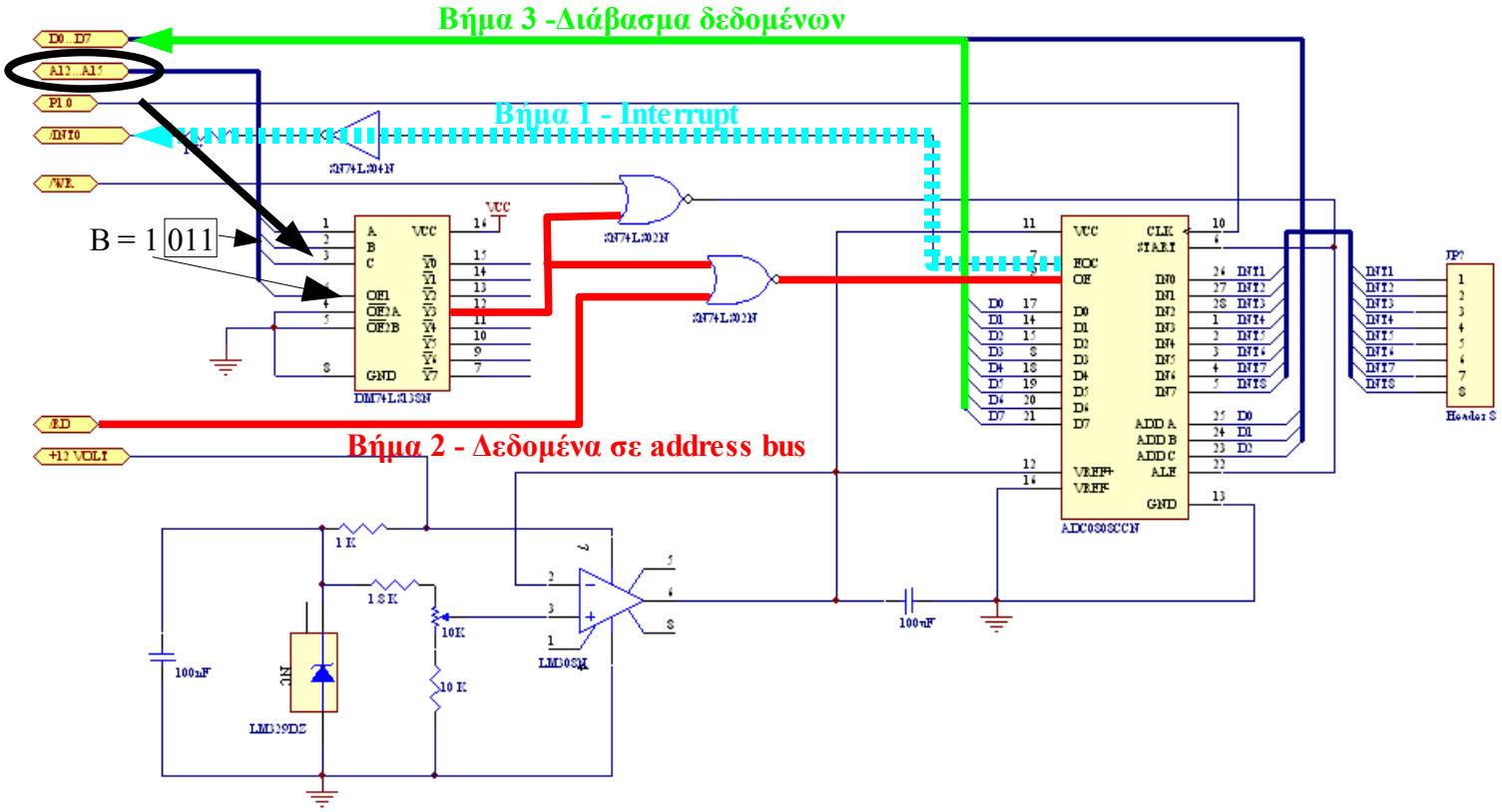
```
int uchar2int(unsigned char x) {
        int i; i=(int)x-48;
        return i;
}
```

Μετατρέπει έναν χαρακτήρα από 0-9, από το ASCII σύστημα σε ακέραιο.

```
void ReadADCData(void) interrupt 0 using 1 {
        ADC.value = ADCrd;
        ADC_Busy = 0;
}
```

Διάβασε την τιμή της ADCrd που βρίσκεται στην θέση μνήμης 0xB000 και αποθήκευσέ την στη μεταβλητή ADC.value. Όταν διαβαστεί αυτή η τιμή, θέτει την τιμή ADC_Busy σε μηδέν, δηλώνοντας ότι την διάβασε.

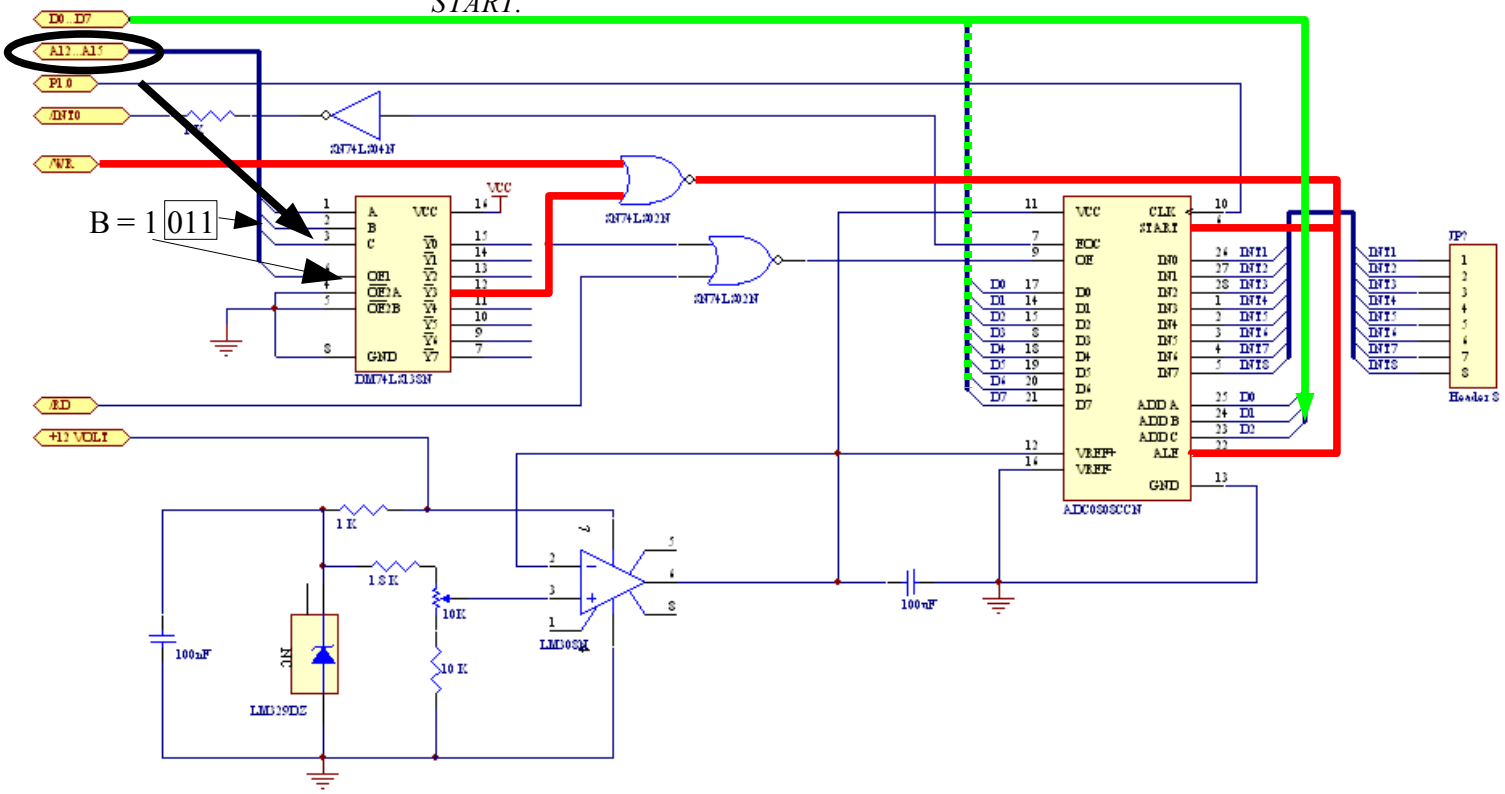
Διαβάζει τα δεδομένα του ADC, όταν γίνει Interrupt.



Σημειώση: *το σήμα OE ενεργοποιεί την έξοδο των δεδομένων D0…D7 στους ακροδέκτες του μετατροπέα*

```
void SetADCInput(unsigned char in) {
        ADC.input = in;
        ADCrd = in;
        ADC_Busy = 1;
}
```

Αποθηκεύω την τιμή της in στην ADC.input, και την στέλνω στο κύκλωμα για να θέσω συγκεκριμένο AD για μετατροπή AD δεδομένων. Τα 3 LSB της in καθορίζουν τον AD. Θέτω ADC_Busy=1, γιατί ξεκινάω την μετατροπή ενεργοποιώντας ταυτόχρονα το START.



Σημείωση: *Επιλέγεται μία από τις αναλογικές εισόδους του μετατροπέα την στιγμή μετάβασης του σήματος ALE από "0" σε "1"*

Η παρακάτω συνάρτηση καλείται μια φορά για την αρχικοποίηση των απαραίτητων χρονιστών/Interrupts.

```
void InitADC(void) {
        C_T2=0;                         //Init Clock from T2
        T2MOD=2;                        //Init Clock from T2
        RCAP2H=0xFF;                    //500KHz
        RCAP2L=0xFA;                    //500KHz
        TR2=1;                           //Timer 2 Start
        //----------------------
        IT0=1; //Edge trigered interruption on (INT0)
        EX0 = 1; // Set on /INT0 for ADC
        //----------------------
}
```

```
void main(void) {
	unsigned int x=0;
	SCON = 0x50;
	TMOD |= 0x20;
	TH1 = 0xf4;		//Set BaudRate at 2400b/s (11.0592MHz clock)
	TR1 = 1;
	TI = 1;
	InitADC();		//Initialization of ADC
	EA=1;			//All interupts enabled.

	while(1) {
		printf ("Insert the number of the analog input\n");
		while((x=uchar2int(getchar()))>8 || x<0) {
			printf ("\n the number must be between 1 and 8 \n");
		}
		SetADCInput(x);
		while(ADC_Busy);
		printf ("\n AN %d: %d\n",(unsigned int)ADC.input,(unsigned int)ADC.value);
	}
}
```

Επιλέγω την αντίστοιχη είσοδο για τον AD, και ξεκινάει η διαδικασία μετατροπής.
Περιμένω μέχρι να γίνει η μετατροπή. Όταν καλείται η συνάρτηση ReadADCdata() μέσω Interrupt, τα δεδομένα αποθηκεύονται στην μεταβλητή ADC.value και η ADC_Busy γίνεται 0. Έτσι φεύγω από τον βρόγχο while.

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.

1)

```
#include <AT898252.H>
#include <stdio.h>
#include <ctype.h>
```

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

2)

```
unsigned char ADCrd; //_at_ 0xB000;

struct record {
	unsigned char input;
	unsigned char value;
};

xdata struct record ADC;
bit char ADC_Busy = 1;
```

α) Δεν δηλώνω την θέση μνήμης του ADCrd. Αφού ξέρω τι κάνει αυτή η μεταβλητή και πως το κάνει, μπορώ να την εξομοιώσω χειροκίνητα/νοητά.

β) Αφαιρώ την C51 έκφραση xdata. Η μεταβλητή (δομή) θα εξομοιωθεί σε τυχαία θέση του PC μου, άρα δεν χρειάζεται αυτός ο ορισμός.

γ) Αντικαθιστώ την C51 λέξη bit, με την ANCI C λέξη char. Αφού το πόσα bits χρησιμοποιεί η ADC_Busy δεν με ενδιαφέρει (παρά μόνο οι τιμές τις που είναι 0 ή 1) μπορώ άνετα να την κάνω 8-bit μεταβλητή.

3)

```
int uchar2int(unsigned char x) {
	int i; i=(int)x-48;
	return i;
}
```

Παραμένει ως έχει. Είναι σε μορφή ANSI C.

4)

```
void ReadADCData(void) { //interrupt 0 using 1 {
	ADC.value = ADCrd;
	ADC_Busy = 0;
}
```

Από συνάρτηση διακοπής, την κάνω απλή συνάρτηση. Κατά την εξομοίωση θα την καλώ εγώ σαν να κάνω εγώ την διακοπή εκεί που χρειάζεται.

5)

```
void SetADCInput(unsigned char in) {
	ADC.input = in;
	ADCrd = in;
	ADC_Busy = 1;
}
```

Παραμένει ως έχει. Είναι σε μορφή ANCI C.

6)

```
void InitADC(void) {
	C_T2=0;                         //Init Clock from T2
	T2MOD=2;                        //Init Clock from T2
	RCAP2H=0xFF;                    //500KHz
	RCAP2L=0xFA;                    //500KHz
	TR2=1;                           //Timer 2 Start
	//-------------------
	IT0=1; //Edge trigered interruption on (INT0)
	EX0 = 1; // Set on /INT0 for ADC
	//-------------------
```

```
}
```

Η συνάρτηση αυτή αναφέρεται σε αρχικοποίηση τιμών που σχετίζονται με Hardware. Στην εξομοίωση αυτή η αρχικοποίηση δεν χρειάζεται - καθώς εκεί δεν υπάρχει "το hardware".

7)

```
void main(void) {
	unsigned int x=0;
	SCON = 0x50;
	TMOD |= 0x20;
	TH1 = 0xf4;		//Set BaudRate at 2400b/s (11.0592MHz clock)
	TR1 = 1;
	TI = 1;
	InitADC();		//Initialization of ADC
	EA=1;			//All interupts enabled.

	while(1) {
		printf ("Insert the number of the analog input\n");
		while((x=uchar2int(getchar()))>8 || x<0) {
			printf ("\n the number must be between 1 and 8 \n");
		}
		SetADCInput(x);
		while(ADC_Busy) { // ;
			ReadADCData();
		}
		printf ("\n AN %d: %d\n",(unsigned int)ADC.input,(unsigned int)ADC.value);
	}
}
```

α) Οι μεταβλητές που σχετίζονται με την αρχικοποίηση τιμών για Hardware δεν χρειάζονται.

β) Μέσα στην while καλείται πλέον η συνάρτηση που θα την καλούσε προηγουμένως η διαδικασία interrupt.

Υπάρχει μια "λογική" διαφορά πλέον - μέσα στην while() η ReadADCData() καλείται σε κάθε επανάληψη, ενώ μέσω interrupt θα καλούταν μόνο μια φορά σε έναν αριθμό επαναλήψεων. Ουσιαστικά όμως και όσων αφορά την εξομοίωση δεν υπάρχει διαφορά, αφού στην ουσία θέλουμε να καλεστεί η συνάρτηση και να βγούμε από την while(), χωρίς να μας ενδιαφέρει πόσες φορές θα καλεστεί η while() πρωτού καλεστεί η ReadADCData().

Πλέον το πρόγραμμα είναι σε μορφή ANSI C, και μπορεί να εξομοιωθεί σε οποιοδήποτε IDE. Η εξομοίωση γίνεται μέσω της διαδικασίας του Debugging, ελέγχοντας κάθε φορά τα αποτελέσματα αν είναι λογικά.

Τελικός κώδικας για εξομοίωση:

```
#include <stdio.h>
#include <ctype.h>

unsigned char ADCrd;

struct record {
	unsigned char input;
	unsigned char value;
};

struct record ADC;
char ADC_Busy = 1;

int uchar2int(unsigned char x) {
	int i; i=(int)x-48;
	return i;
}

void ReadADCData(void) {
	ADC.value = ADCrd;
	ADC_Busy = 0;
}

void SetADCInput(unsigned char in) {
	ADC.input = in;
	ADCrd = in;
	ADC_Busy = 1;
}

void main(void) {
	unsigned int x=0;
	while(1) {
		printf ("Insert the number of the analog input\n");
		while((x=uchar2int(getchar()))>8 || x<0) {
			printf ("\n the number must be between 1 and 8 \n");
		}
		SetADCInput(x);
		while(ADC_Busy) {
			ReadADCData();
		}
		printf ("\n AN %d: %d\n",(unsigned int)ADC.input,(unsigned int)ADC.value);
	}
}
```