ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Εργαστήριο
Μικροελεγκτές - Ενσωματωμένα Συστήματα

2ος ΚΥΚΛΟΣ

Μικροελεγκτές Atmel – AVR

ΑΣΚΗΣΗ 3η

Εργαλεία Ανάπτυξης εφαρμογών για μικροελεγκτές AVR σε γλώσσα C – Λειτουργίες I/O

Γ. Καλτσάς
Καθηγητής

Αθήνα 2019

1. Προγραμματισμός σε γλώσσα C – Εργαλεία για AVR

1.1 Εισαγωγή

Η γλώσσα προγραμματισμού Assembly είναι μια γλώσσα προγραμματισμού χαμηλού επιπέδου. Ο όρος "γλώσσα προγραμματισμού χαμηλού επιπέδου", σημαίνει ότι το σύνολο των εντολών της είναι ένα βήμα πιο πάνω από την γλώσσα μηχανής η οποία αποτελείται από 0 (μηδέν) και 1 (ένα) πού είναι και το μόνο που μπορεί να καταλάβει ένας επεξεργαστής, για την εκτέλεση του προγράμματος. Υπεύθυνο εργαλείο για να μετατραπεί ένα πρόγραμμα γραμμένο σε γλώσσα Assembly, σε γλώσσα μηχανής για τον επεξεργαστή, είναι ο *Assembler* ή *Συμβολομεταφραστής*. Όπως παρατηρήθηκε όμως, ακόμα και αυτή η γλώσσα προγραμματισμού δυσκολεύει αρκετά τον προγραμματιστή στην σύνταξη ενός προγράμματος, για αυτό τον λόγο δημιουργήθηκαν οι γλώσσες ανώτερου επιπέδου όπως π.χ. C++, Python, Java κ.α.. Οι γλώσσες προγραμματισμού ανώτερου επιπέδου είναι σχεδιασμένες για να είναι φιλικές στον προγραμματιστή καθώς μοιάζουν με περιορισμένη φυσική γλώσσα. Παρόμοια όπως και στην γλώσσα προγραμματισμού Assembly, χρειαζόμαστε ένα εργαλείο να για μεταφράσει το πρόγραμμα γραμμένο σε γλώσσα ανώτερου επιπέδου σε γλώσσα μηχανής. Αυτό επιτυγχάνεται από τον *Μεταγλωττιστή* ή *Compiler*. Ο Μεταγλωττιστής ή Compiler είναι ένα πρόγραμμα που έχει είσοδο το πρόγραμμα γραμμένο σε ανώτερη γλώσσα και έξοδο ένα δυαδικό *αρχείο* ***.hex*** (στην περίπτωση των μικροεπεξεργαστών AVR) που είναι εκτελέσιμο από τον επεξεργαστή.

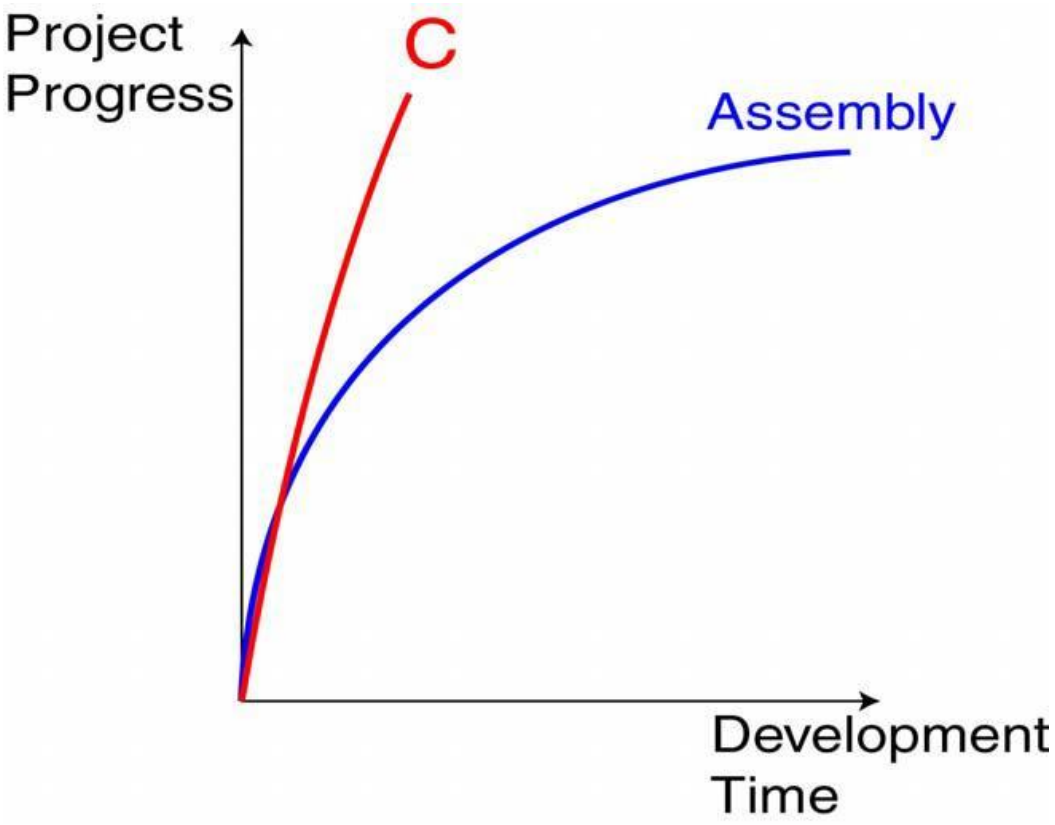
Η γλώσσα προγραμματισμού C δημιουργήθηκε από τον Dennis Richie και άλλους συναδέλφους του στις αρχές της δεκαετίας του 1970, στα εργαστήρια *Bell Labs* της εταιρείας *AT&T*. Εκείνη την εποχή, ο κώδικας της πρώτης έκδοσης του λειτουργικού συστήματος *Unix*, που επίσης είχε αναπτυχθεί στα *Bell Labs* το 1969, ήταν γραμμένος σε assembly. Ο D. Richie αποφάσισε να τον ξαναγράψει σχεδόν ολοκληρωτικά σε μία γλώσσα υψηλότερου επιπέδου, ώστε ο έλεγχος της λειτουργικότητάς του, καθώς και μελλοντικές τροποποιήσεις ή αναβαθμίσεις του, να γίνονται με ευκολότερο τρόπο. Η ανάγκη για την προτυποποίηση της γλώσσας C οδήγησε το 1983 τον οργανισμό American National Standard Institute *(ANSI)* να ορίσει μία επιστημονική επιτροπή για την ανάπτυξη ενός διεθνούς προτύπου. Αυτή η προσπάθεια ολοκληρώθηκε το 1989 με τη δημιουργία ενός προτύπου, το οποίο περιγράφει με σαφήνεια τα χαρακτηριστικά της γλώσσας (*ISO/IEC 9899:1990*). Το πρότυπο αυτό έχει καθιερωθεί ως ANSI C ή Standard C και είναι υποχρεωτικό να υποστηρίζεται από όλους τους C μεταγλωττιστές.
Οι μικροελεγκτές AVR είναι σχεδιασμένοι ώστε σε συνδυασμό με τα απαραίτητα εργαλεία (toolchain), να είναι συμβατοί και να εκτελούν ένα μεγάλο ποσοστό κώδικα προτυποποιημένου κατά ANSI C. Ένα ενδεικτικό toolchain συνοπτικά:

- avr-gcc (ο compiler που αναλαμβάνει την μεταγλώττιση του κώδικα C σε γλώσσα μηχανής),
- avr-libc (απαραίτητη βασική βιβλιοθήκη που ενσωματώνει πολλές από τις συναρτήσεις της Standard C για AVR, καθώς κι αρκετές συναρτήσεις αποκλειστικά ανεπτυγμένες για τους μικροελεγκτές αυτούς).

Τα πλεονεκτήματα ανάπτυξης κώδικα σε C σε σχέση με προγραμματισμό σε assembly είναι (συνοπτικά):

✓ Μέχρι και 50% μικρότερος κώδικας σε σχέση με άλλες αρχιτεκτονικές,
✓ Ευκολία εκμάθησης,
✓ Χρήση σε διαφορετικά συστήματα μικροελεγκτών με μικρές διαφοροποιήσεις,

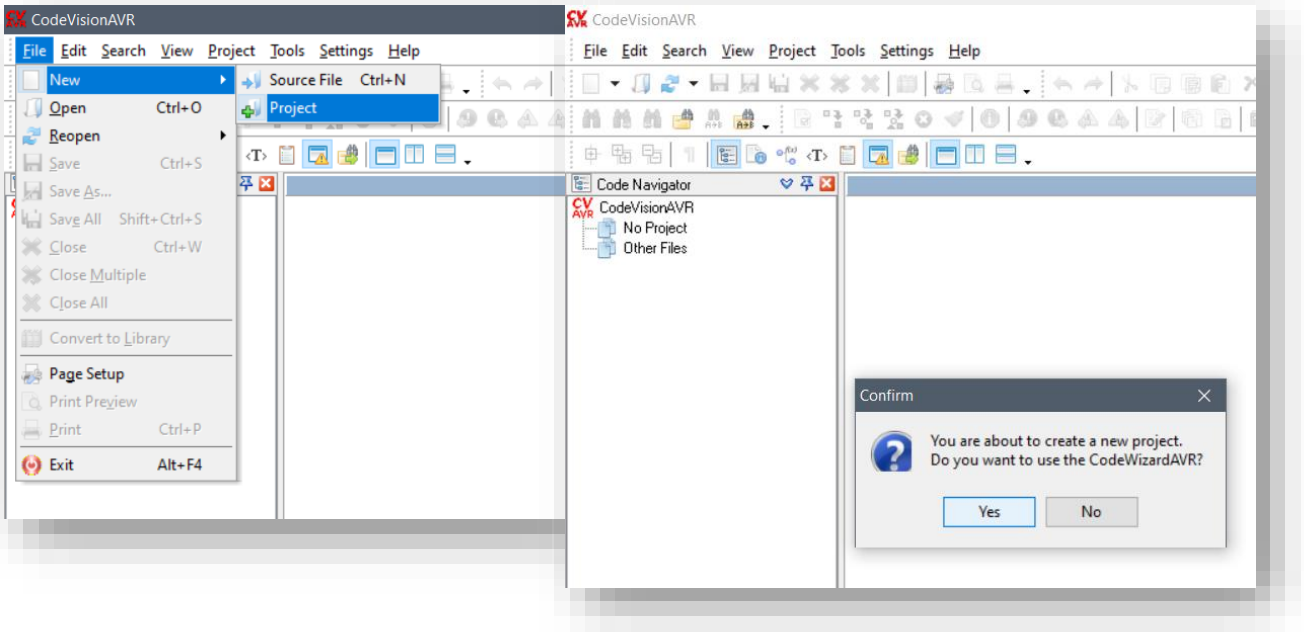
- ✓ Συντήρηση και αναβάθμιση του κώδικα με μικρότερο κόστος και εργατοώρες μηχανικού,
- ✓ Επαναχρησιμοποίηση κώδικα σε διάφορα project με μικρές ή χωρίς αλλαγές (βιβλιοθήκες με ρουτίνες, συναρτήσεις κλπ.)



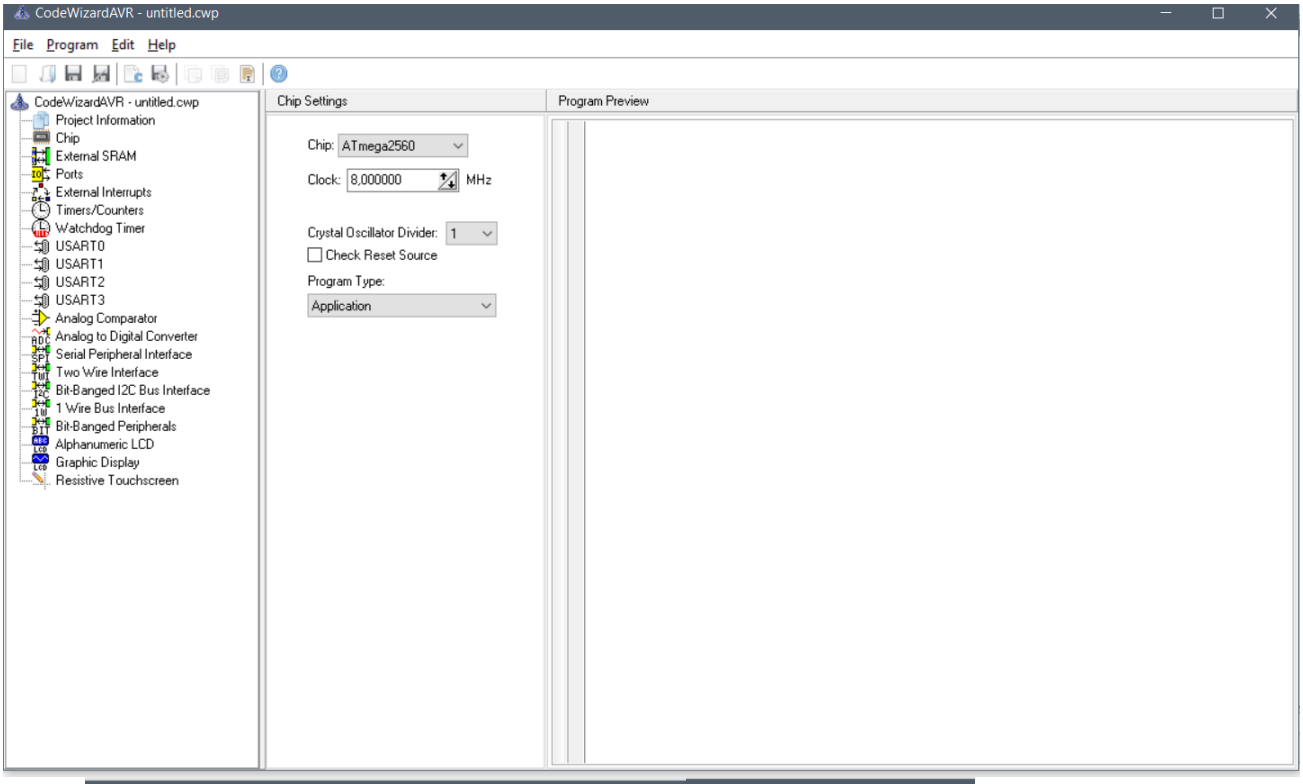
2 AVR CodeVision – CodeWizard

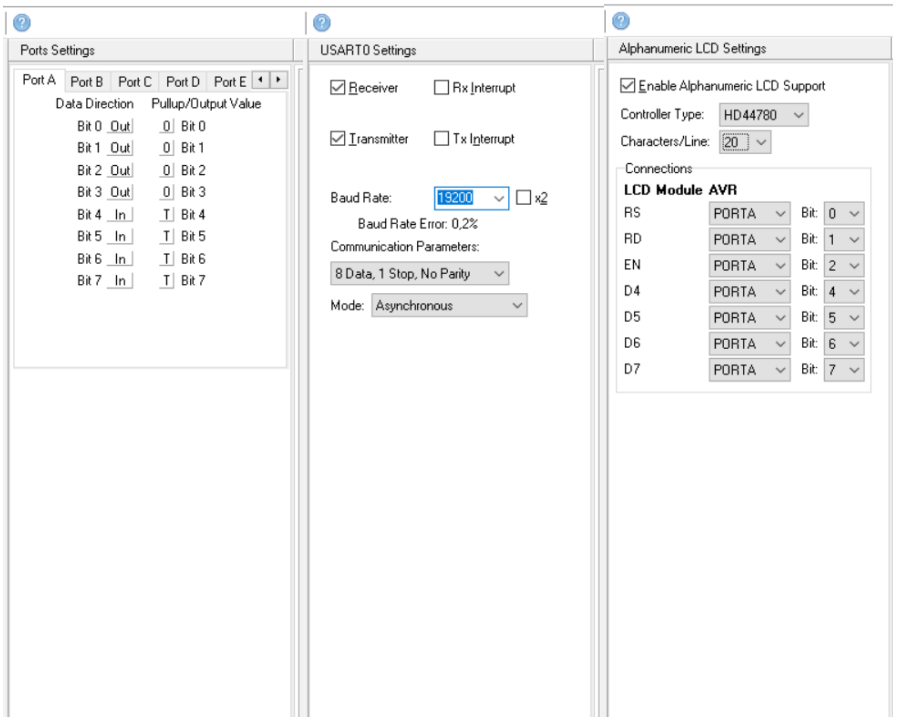
Το λογισμικό AVR CodeVision είναι ένα ενοποιημένο περιβάλλον ανάπτυξης και αυτοματοποιημένης παραγωγής κώδικα (IDE, *Integrated Development Environment*) για την οικογένεια μικροελεγκτών AVR. Συμπεριλαμβάνει όλα τα απαραίτητα εργαλεία για την συγγραφή κώδικα (code editor), την μεταγλώττιση κι αποσφαλμάτωση (compiler), την μεταφορά του εκτελέσιμου προγράμματος στον μικροελεγκτή, καθώς και βιβλιοθήκες για περιφερειακά όπως οθόνες lcd. Επίσης, με το εργαλείο CodeWizardAVR, ο χρήστης μπορεί να επιλέξει τις συνθήκες και τα περιφερειακά που επιθυμεί να χρησιμοποιήσει κι ο κώδικας παράγεται αυτόματα σύμφωνα με τις επιλογές αυτές.

2.1 Δημιουργία Project με CodeWizardAVR



Με την εντολή File → New → Project κι επιλέγοντας «Yes» στην ερώτηση για την χρήση του CodeWizardAVR, ο χρήστης μπορεί στην συνέχεια να αρχικοποιήσει τα επιθυμητά περιφερειακά.
Στις παρακάτω εικόνες, εμφανίζεται ένα ενδεικτικό παράδειγμα χρήσης του εργαλείου CodeWizardAVR. Χρησιμοποιώντας το συγκεκριμένο εργαλείο επιλέγουμε αρχικά τον τύπο (ATmega2560) και τον χρονισμό του μικροελεγκτή (8 MHz). Στη συνέχεια επιλέγουμε το είδος των περιφερειακών που θέλουμε να χρησιμοποιήσουμε (πόρτες, σειριακή επικοινωνία μέσω της USART0, αλφαριθμητική οθόνη LCD). Τέλος επιλέγουμε τον τρόπο (αρχικοποίηση) που θέλουμε να λειτουργούν τα συγκεκριμένα περιφερειακά. Ειδικότερα επιλέγουμε ποιες πόρτες θα χρησιμοποιήσουμε και αν θα λειτουργήσουν ως είσοδοι ή ως έξοδοι, τις παραμέτρους της σειριακής επικοινωνίας USART και τις παραμέτρους της αλφαριθμητικής LCD οθόνης που θέλουμε να χρησιμοποιήσουμε.

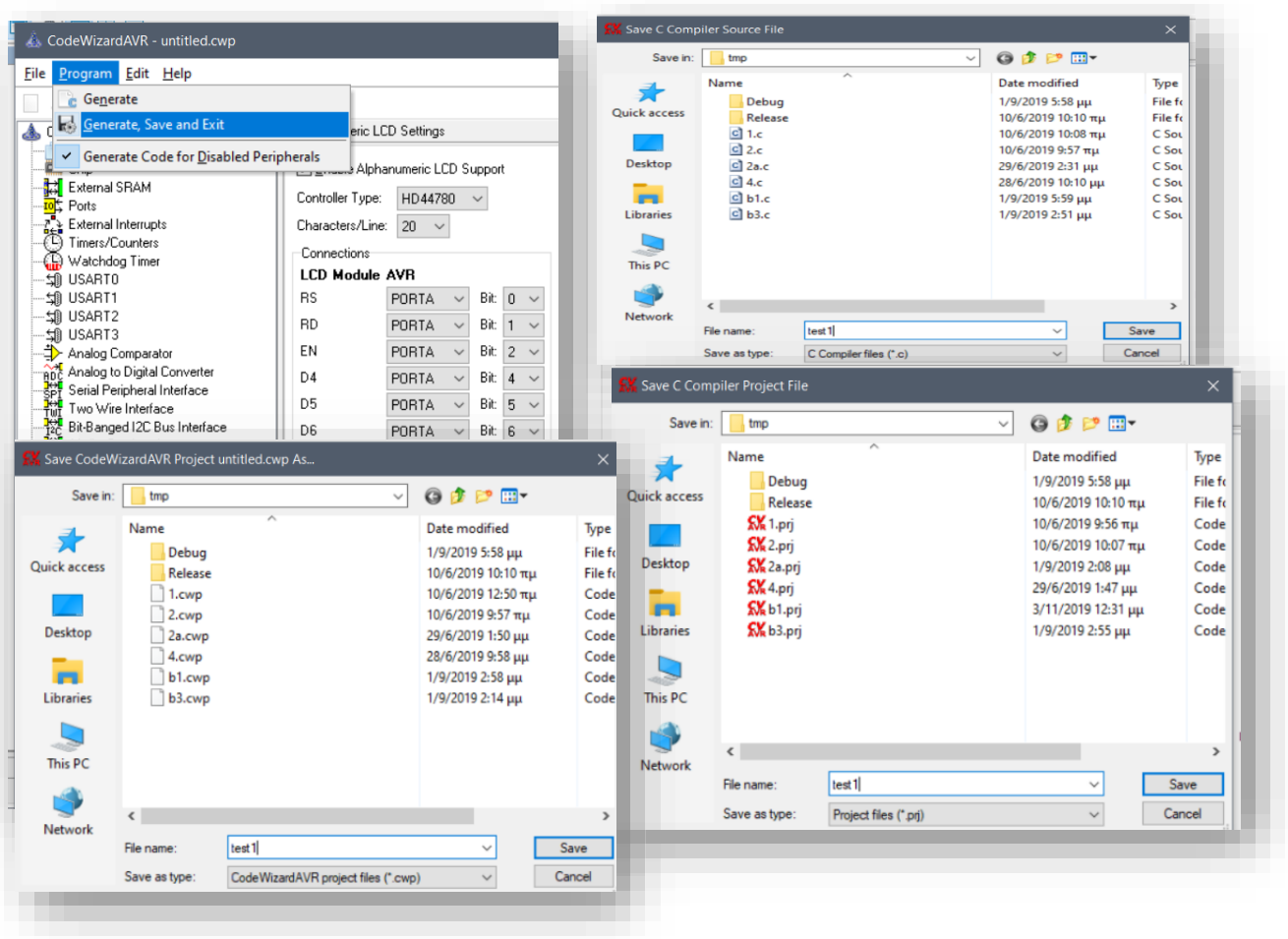




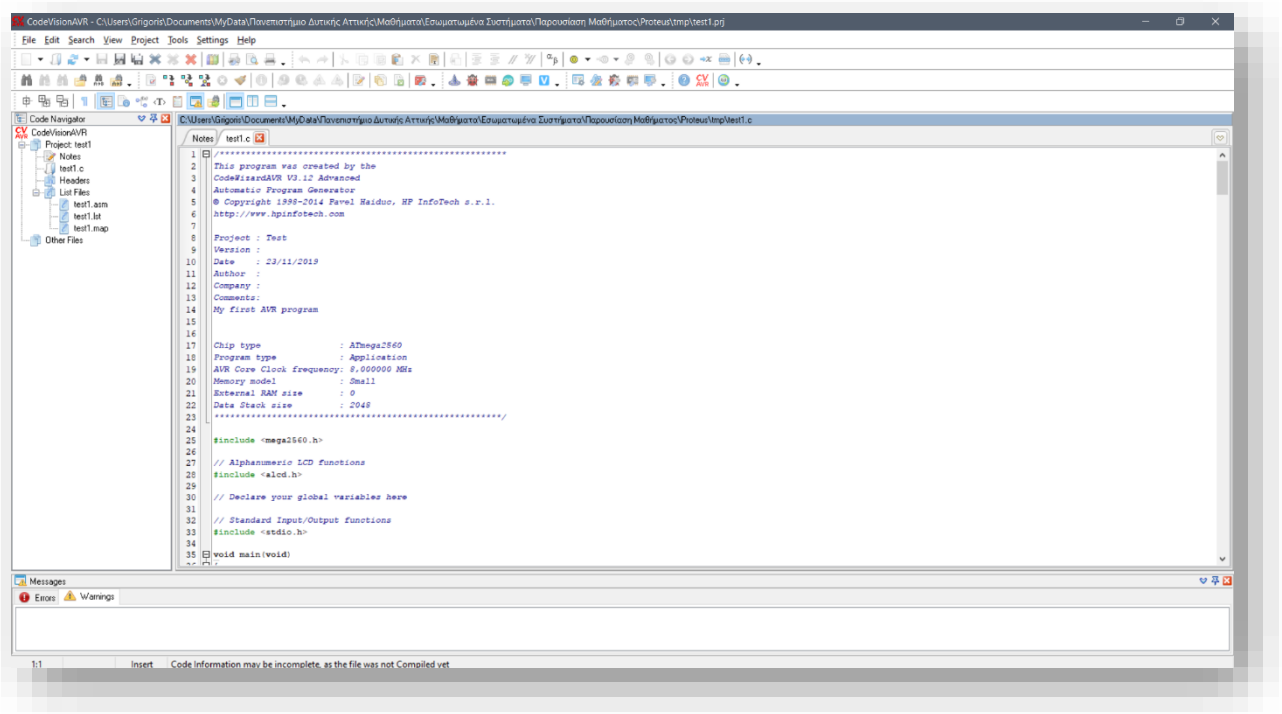
Μετά το τέλος των ρυθμίσεων, ο χρήστης πρέπει να εκτελέσει τις ακόλουθες ενέργειες για να παραχθεί ο απαραίτητος κώδικας αρχικοποίησης:

Στο παράθυρο του CodeWizard:
Program → Generate, Save & Exit. Τέλος, ζητείται από τον χρήστη να ονομάσει τα αρχεία που θα δημιουργηθούν (ένα αρχείο .prj, που είναι το αρχείο project του

CodeVision, ένα αρχείο .cwp που είναι το CodeWizard project κι ένα αρχείο .c με τον πηγαίο κώδικα σε C). **Τα αρχεία πρέπει να αποθηκευτούν με το ίδιο όνομα, π.χ. test_project.prj, test_project.cwp & test_project.c**



Στο σημείο αυτό, τα περιφερειακά είναι ρυθμισμένα κατά τις απαιτήσεις του χρήστη και ξεκινάει η συγγραφή του κώδικα στο αρχείο που έχει ανοίξει στο παράθυρο του editor.



Αφού ολοκληρωθεί η ανάπτυξη του κώδικα, το επόμενο βήμα είναι η μεταγλώττιση σε γλώσσα μηχανής (Project → Build/Build All). Το CodeVision παρέχει τις απαραίτητες πληροφορίες για το αποτέλεσμα της μεταγλώττισης. Ο μεταγλωττιστής υποδεικνύει ενδεχόμενα λάθη ή προειδοποιήσεις (warnings) στο ειδικό παράθυρο “Messages”. Αν δεν υπάρχουν λάθη, ο μεταγλωττιστής παράγει το εκτελέσιμο αρχείο (π.χ. με επέκταση .cof).

1.2.2 Περίγραμμα προγράμματος

Όλα τα προγράμματα για τους μικροελεγκτές AVR (και για αρκετούς ακόμη με μικρές αλλαγές αλλά με την ίδια λογική), ακολουθούν μια προκαθορισμένη δομή:

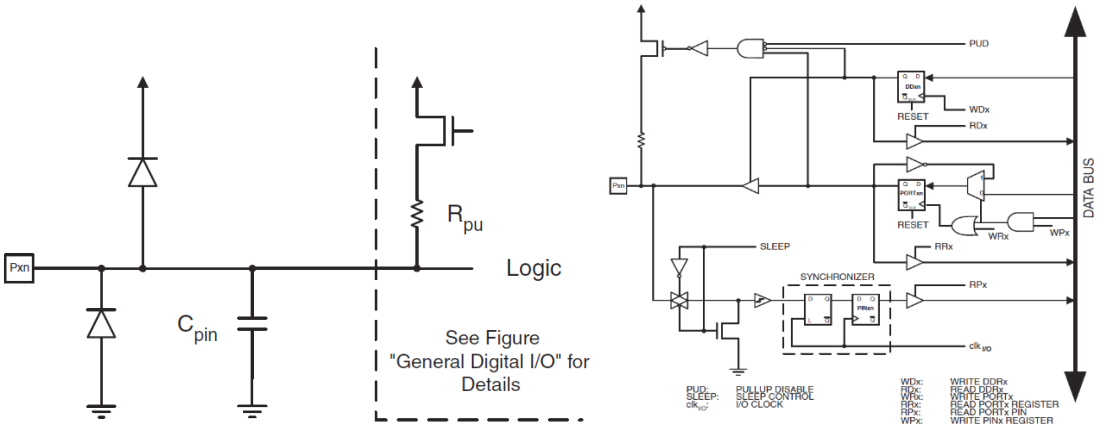
```
✓ Δηλώσεις συμπεριλαμβανομένων αρχείων     (include files)
✓ Δηλώσεις γενικών μεταβλητών               (global variables)
✓ Δηλώσεις συναρτήσεων                      (functions)

void main(void)
{
    - Δηλώσεις τοπικών μεταβλητών      (local variables)

    - Αρχικοποίηση                     (initialization):

        Εισόδων – Εξόδων               (I/O)
        Χρονιστών – Μετρητών           (Timers - Counter)
        Εξωτερικών διακοπών            (External Interrupts)
        Διακοπών Χρονιστών - Μετρητών
        Μονάδας ασύγχρονης επικοινωνίας    (UART)
        Αναλογικού συγκριτή
```

```
        Μετατροπέα αναλογικού σε ψηφιακό   (ADC)
        κ.α.  ...

    while (1)
    {

        Κώδικας προγράμματος

    }
}
```

Δηλαδή, μετά από την εκτέλεση των αρχικών εντολών για προσθήκη βιβλιοθηκών, ορισμούς συναρτήσεων, μεταβλητών και αρχικοποίηση περιφερειακών, ο μικροελεγκτής «παγιδεύεται» σε έναν ατέρμονα βρόγχο, μέσα στον οποίο εκτελεί την εργασία για την οποία τον προγραμματίζει ο χρήστης.

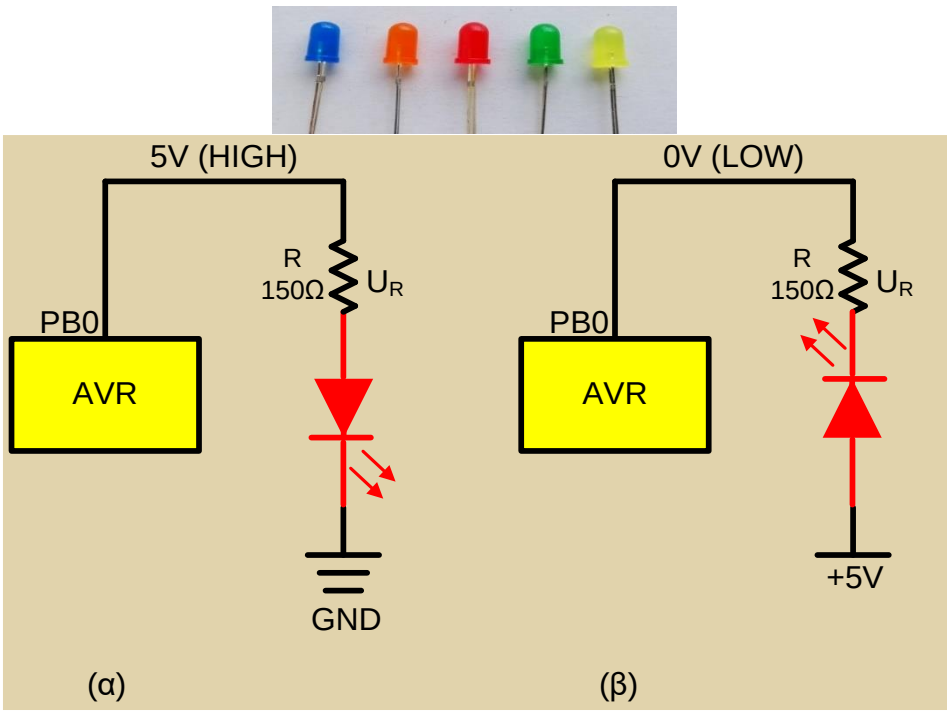
```
#include <mega2560.h>   // include files
#include <delay.h>      // include files

void main(void)
{
    int i;              // δήλωση τοπικών μεταβλητών

/* αρχικοποιήσεις */
    // Crystal Oscillator division factor: 1
    #pragma optsize-
    CLKPR=0x80;
    CLKPR=0x00;
    #ifdef _OPTIMIZE_SIZE_
        #pragma optsize+
    #endif

    PORTB = 0x00;   // αρχικοποίηση PORTB
    DDRB = 0xFF;
/* /αρχικοποιήσεις */

    while (1)   /* κώδικας προγράμματος */

    {
        for(i=0;i<256;i++)
        {
            PORTB=i;
            delay_ms(500);
        }
    }               /* /κώδικας προγράμματος */
}
```

2 Πόρτες Εισόδου/Εξόδου



- ✓ Κάθε ακροδέκτης ψηφιακής εισόδου/εξόδου (Digital I/O) έχει εσωτερικά διόδους προστασίας από υπερτάσεις (clamp diodes).
- ✓ Κατά την λειτουργία εισόδου, υπάρχει εσωτερικά μια αντίσταση pull-up, η οποία ενεργοποιείται μέσω του κώδικα.
- ✓ Σε περίπτωση reset, οι πόρτες τίθενται σε κατάσταση tri-state.

Συνδεσμολογίες Διατάξεων Εξόδου (LEDs)



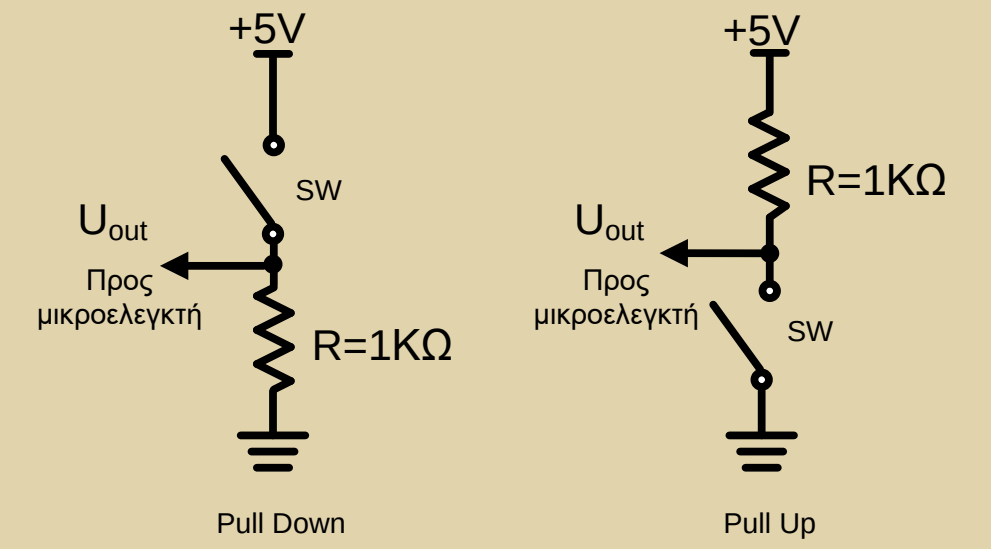
Υπάρχουν δυο βασικοί τρόποι σύνδεσης μιας διόδου LED (ή οποιουδήποτε άλλου φορτίου) με έναν ακροδέκτη σε μικροελεγκτή AVR: a) source & b) sink. Με άλλα λόγια, στην πρώτη περίπτωση παρέχει ο μικροελεγκτής το απαραίτητο ρεύμα στο φορτίο, ενώ

στην δεύτερη είναι υπεύθυνος να συνδέει το φορτίο με την γείωση του κυκλώματος. Θα πρέπει ο χρήστης σε κάθε περίπτωση να μεριμνά ώστε να μην ξεπερνά τα ρεύματα για τα οποία είναι σχεδιασμένοι να λειτουργούν οι ακροδέκτες. Στην σχετική εικόνα παρίστανται οι δύο συνδεσμολογίες, στις οποίες μια δίοδος συνδέεται στον ακροδέκτη PB0 ενός μικροελεγκτή AVR. Στην πρώτη περίπτωση (α) έχουμε συνδεσμολογία θετικής λογικής, όπου η δίοδος ανάβει με κατάσταση «1» στον ακροδέκτη PB0. Στην δεύτερη περίπτωση (β) έχουμε συνδεσμολογία αρνητικής λογικής, όπου η δίοδος ανάβει με κατάσταση «0» στον ακροδέκτη PB0.

2.2 Συνδεσμολογίες Διατάξεων Εισόδου (Διακόπτες)

Οι μικροελεγκτές, όπως έχει αναφερθεί έχουν ακροδέκτες που μπορούν να χρησιμεύσουν σαν Είσοδοι και σαν Έξοδοι *(Inputs/Outputs I/O)* και συνήθως ονομάζονται γενικής χρήσεως Είσοδοι/Έξοδοι *(General Purpose Input/Output GPIO)*. Αυτοί οι ακροδέκτες μπορούν να βρίσκονται σε τρεις καταστάσεις
1) Λογική Στάθμη [1] με δυναμικό 5V .
2) Λογική Στάθμη [0] με δυναμικό 0V .
3) Καμία Λογική Στάθμη [Υψηλής Εμπέδησης **ή** Floating **ή** Tri-stated] με δυναμικό άγνωστο για τον ακροδέκτη.
Είναι σημαντικό να λειτουργούμε τους ακροδέκτες του μικροελεγκτή σε γνώστη λογική στάθμη (0 ή 1), με Pull-Up ή Pull-Down αντιστάσεις.

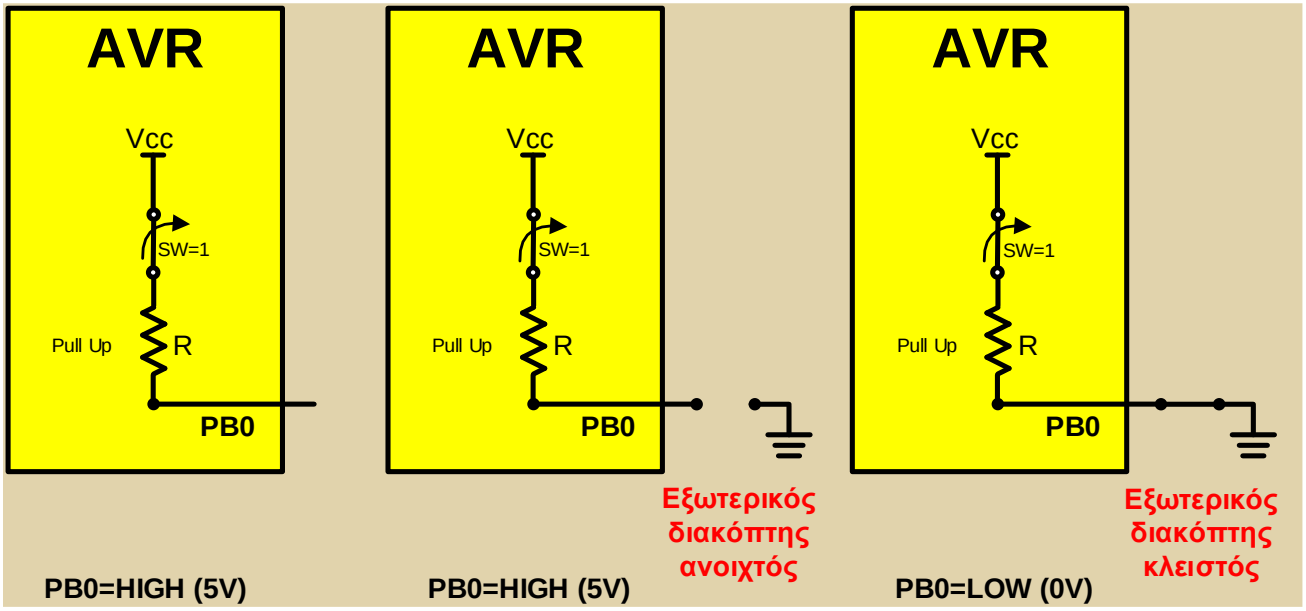
Στο παρακάτω σχήμα απεικονίζονται οι δύο συνδεσμολογίες σύνδεσης ψηφιακών διατάξεων εισόδου σε ακροδέκτες μικροελεγκτή:



Λειτουργία κυκλωμάτων διακοπτών					
Pull Down			Pull Up		
SW	Uout	Λογική στάθμη	SW	Uout	Λογική στάθμη
A	0V	0	A	5V	1
K	5V	1	K	0V	0
A = Ανοιχτός διακόπτης, K = Κλειστός διακόπτης					

2.3 Συνδεσμολογίες Διατάξεων Εισόδου σε μC AVR

Ο μικροελεγκτής AVR, όπως θα δούμε και παρακάτω, έχει εσωτερικές Pull-Up Αντιστάσεις οπότε μας δίνει την δυνατότητα να χρησιμοποιήσουμε τα περιφερειακά μας σε απευθείας συνδεσμολογία.



Οι πόρτες των 8-bit μικροελεγκτών είναι οργανωμένες ανά οκτώ ακροδέκτες, όπου ο καθένας από αυτούς μπορεί να διαμορφώνεται και να λειτουργεί ανεξάρτητα από τους υπόλοιπους. Η διαμόρφωση κάθε πόρτας του μικροελεγκτή καθορίζεται από τρεις καταχωρητές:

α. `DDRx`. Καθορίζει την λειτουργία των ακροδεκτών της πόρτας `x`, δηλαδή, αν θα είναι είσοδοι ή έξοδοι. Αν η τιμή κάποιου bit είναι 1 ο αντίστοιχος ακροδέκτης καθορίζεται σαν έξοδος, ενώ, αν είναι 0, καθορίζεται σαν είσοδος. Για παράδειγμα, με την εντολή:

```
DDRA = 0x00;  // (0b0000000 ή 0)
```

Όλοι οι ακροδέκτες της PORTA δηλώνονται σαν είσοδοι ενώ με την εντολή:

```
DDRA = 0xFF;  // (0b11111111 ή 255)
```

Όλοι οι ακροδέκτες της PORTA δηλώνονται σαν είσοδοι.

Με την εντολή:

```
DDRA = 0x17;  // (0b00010111 ή 23)
```

οι ακροδέκτες `0`, `1`, `2` και `4` της πόρτας `A` καθορίζονται ως έξοδοι και οι υπόλοιποι ως είσοδοι.

Με τις εντολές

```
DDRA  = 0xFF;  // (0b11111111 ή 255)
DDRB  = 0x00;  // (0b00000000 ή 0)
```

όλοι οι ακροδέκτες της PORTB θέτονται σαν έξοδοι ενώ αυτοί της PORTC σαν είσοδοι.

β. `PORTx`. Καθορίζει τις τιμές των ακροδεκτών που έχουν προγραμματιστεί ως έξοδοι, ενώ για τους ακροδέκτες που έχουν προγραμματιστεί ως είσοδοι καθορίζει αν οι αντίστοιχες συνδεδεμένες pull-up αντιστάσεις είναι ενεργοποιημένες (1) ή όχι (0). Για παράδειγμα, με τις εντολές:

```
DDRB  = 0b11111111; // (0xFF ή 255)
PORTB = 0b11111111; // (0xFF ή 255)
DDRC  = 0b11111111; // (0xFF ή 255)
PORTC = 0b11110000; // (0xF0 ή 255)
```

Όλοι οι ακροδέκτες της PORTB δηλώνονται σαν έξοδοι και οι τιμές τους καθορίζονται στα 5V, ενώ όλοι οι ακροδέκτες της PORTC δηλώνονται σαν έξοδοι αλλά 5V θέτονται μόνο στους ακροδέκτες του high nibble.

Με τις εντολές:

```
DDRB  = 0b11001110; // (0xCE ή 206)
PORTB = 0b00011101; // (0x1D ή 29)
DDRA = 0x00;   // (0b00000000 ή 0)
PORTA = 0x0F;  // (0b0000FFFF ή 15)
```

οι ακροδέκτες `1`, `2`, `3`, `6` και `7` της πόρτας `B` καθορίζονται ως έξοδοι και ο καταχωρητής `PORTB` καθορίζει τις τιμές τους. Συγκεκριμένα, οι τιμές των ακροδεκτών `2` και `3` είναι ένα (π.χ. 5 Volts), ενώ οι τιμές των ακροδεκτών `1`, `6` και `7` είναι μηδέν (0 Volts).

οι ακροδέκτες `0`, `4` και `5` καθορίζονται ως είσοδοι και ο καταχωρητής `PORTB` καθορίζει αν έχουν ενεργοποιηθεί ή όχι οι pull-up αντιστάσεις. Συγκεκριμένα, στους ακροδέκτες `0` και `4` έχουν ενεργοποιηθεί οι αντιστάσεις, ενώ στον ακροδέκτη `5` όχι. Η PORTA ρυθμίζεται σαν είσοδος και ενεργοποιούνται οι pullup μόνο του low nibble.

γ. `PINx`. Περιέχει τις τιμές των ακροδεκτών που έχουν προγραμματιστεί ως είσοδοι. Για παράδειγμα, αν η τιμή του `PINB` είναι `0b00110000` οι τιμές των ακροδεκτών εισόδου `4` και `5` είναι ένα, ενώ η τιμή του ακροδέκτη `0` είναι μηδέν.

Το λογισμικό AVRCodeVision παρέχει την δυνατότητα για την προσπέλαση των ακροδεκτών να χρησιμοποιείται ο τελεστής τελεία (`.`). Για παράδειγμα:

```
DDRA.0  = 1;   /* Ο πρώτος αποδέκτης της πόρτας Α καθορίζεται σαν
έξοδος */
PORTA.0 = 1;   /* Εξαγωγή 5V από τον πρώτο ακροδέκτη της πόρτας Α
*/
if (PINC.7)
{ /* if (PINC & 0x80) */
```

```
   /* Κώδικας που εκτελείται αν το έβδομο pin της C είναι «1» */

}
```

Συνοπτικό παράδειγμα:

```
DDRB   = 0b1100 1110
PORTB  = 0b0001 1101
```

PB0: Είσοδος τάσης με pull-up αντίσταση αντίσταση
PB1: Εξαγωγή τάσης 0V αντίσταση
PB2: Εξαγωγή τάσης 5V
PB3: Εξαγωγή τάσης 5V
PB4: Είσοδος τάσης με pull-up
PB5: Είσοδος τάσης χωρίς pull-up
PB6: Εξαγωγή τάσης 0V
PB7: Εξαγωγή τάσης 0V

Bit Masking

Σε πολλές περιπτώσεις θέλουμε να ρυθμίσουμε μόνο συγκεκριμένους ακροδέκτες ή bit από καταχωρητές, αφήνοντας τα υπόλοιπα bits στην κατάσταση που βρίσκονταν. Π.χ. αν τα PB0…PB4 είναι δεσμευμένα από ένα εξωτερικό περιφερειακό, ενώ εμείς θέλουμε να ελέγξουμε ένα LED στο PB7, δεν είναι καλή τακτική να γράψουμε σε όλη την PORTB, μιας και θα παρέμβουμε στην λειτουργία των ακροδεκτών που ήδη βρίσκονται σε χρήση. Για να θέσουμε λογικό «1» σε συγκεκριμένες θέσεις χρησιμοποιούμε την λογική πράξη OR ($x \mid y$) με τον τελεστή «1».

Π.χ. για να θέσουμε λογικό «1» στα bit7 και bit6 της PORTB χωρίς να πειράξουμε τα υπόλοιπα bit:

```
PORTB = PORTB | 0b1100 0000 (ή 0xC0)
```

Κι εκμεταλλευόμενοι το συντακτικό της γλώσσας C ($x = x + 1 \rightarrow x \mathrel{+}= 1$)

```
PORTB |= 0b1100 0000 (ή 0xC0)
```

Για να θέσουμε λογικό «1» σε όλα τα bit της PORTB εκτός από το 3ο και το 4ο:

```
PORTB |= 0b1111 0011
```

Όμοια, για να θέσουμε σε λογικό «0» διάφορα bit χωρίς να αλλάξουμε την κατάσταση των υπολοίπων, χρησιμοποιούμε την λογική πράξη AND με τον τελεστή «0».

Π.χ. για να θέσουμε λογικό «0» στα bit7 και bit6 της PORTA χωρίς να πειράξουμε τα υπόλοιπα bit:

```
PORTA &= 0b0011 1111
```

Για να θέσουμε λογικό «0» σε όλα τα bit της PORTC εκτός από το LSB:

```
PORTC &= 0b0000 0001
```

Τέλος, για να αντιστρέψουμε την λογική κατάσταση συγκεκριμένων bit χωρίς να αλλάξουμε την κατάσταση των υπολοίπων, χρησιμοποιούμε την λογική πράξη XOR με τον τελεστή «1».

Π.χ. για να αντιστρέψουμε την λογική κατάσταση των bit7 και bit6 της PORTA χωρίς να πειράξουμε τα υπόλοιπα bit:

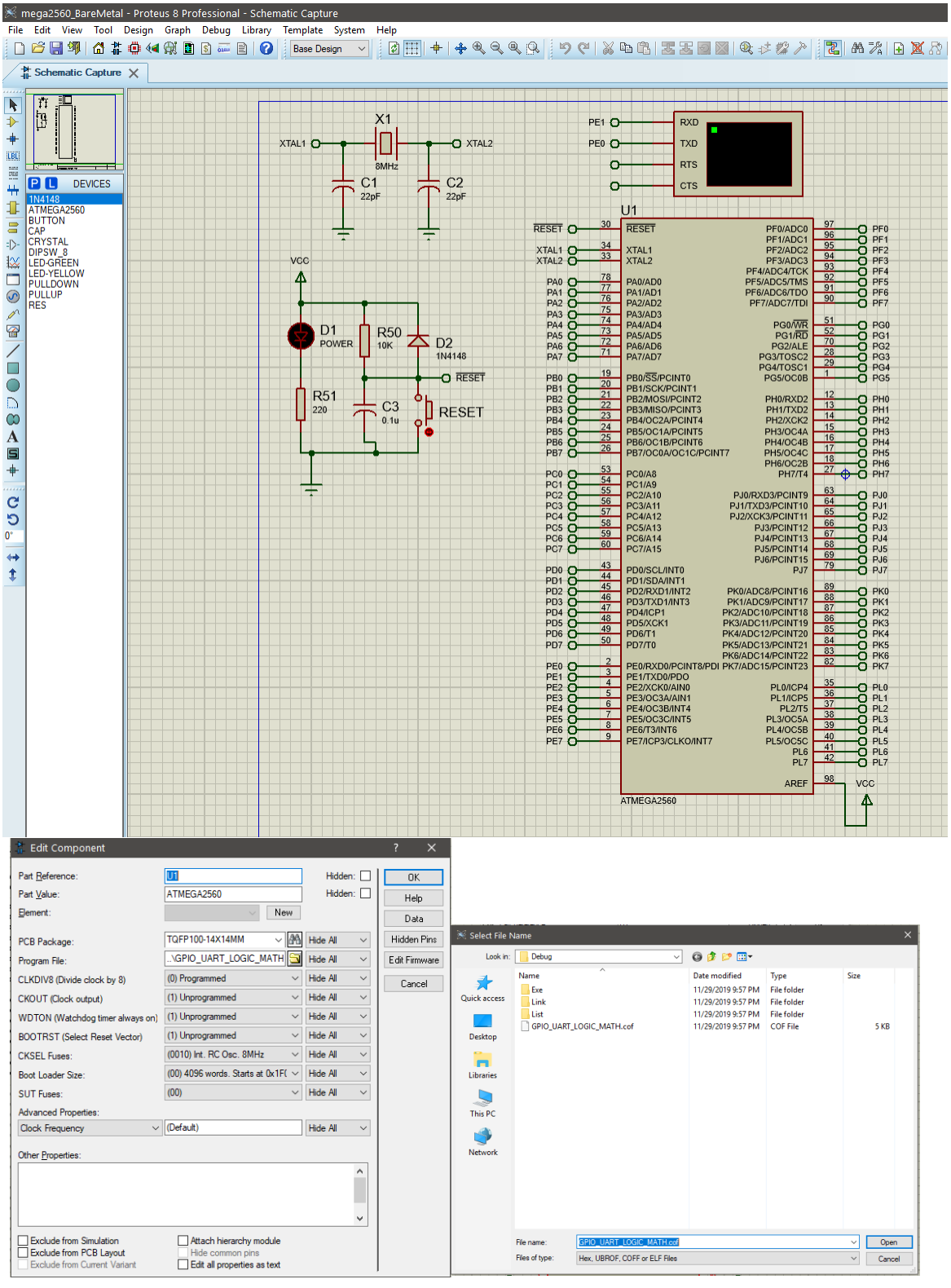
```
PORTA ^= 0b1100 0000
```

Για να αντιστρέψουμε την λογική κατάσταση σε όλα τα bit της PORTC εκτός από το LSB:

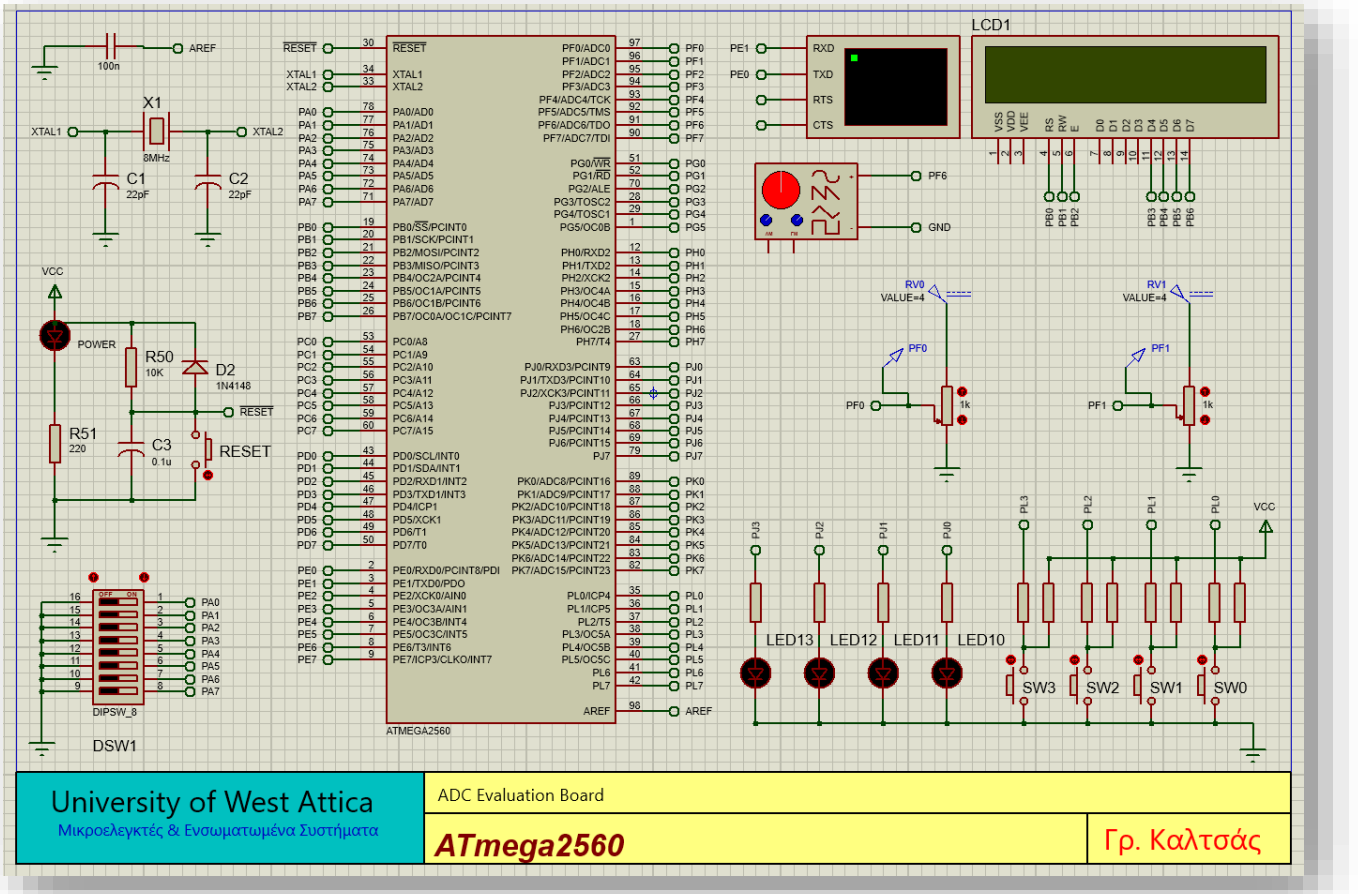
```
PORTC &= 0b1111 1110
```

3 Proteus

Το περιβάλλον Proteus καθιστά δυνατή την εκτέλεση προγραμμάτων για διάφορους μικροελεγκτές AVR σε συνδυασμό με άλλα περιφερειακά και ηλεκτρονικά εξαρτήματα. Τα εκτελέσιμα προγράμματα που αναπτύχθηκαν στα προαναφερθέντα κεφάλαια σε γλώσσα C, μπορούν να ενσωματωθούν και να εκτελεστούν μέσω Proteus.
Για να το πετύχουμε αυτό, φορτώνουμε το εκτελέσιμο αρχείο .cof που παράγεται από το CodeVision. Με διπλό κλικ στον μικροελεγκτή, ανοίγει το παράθυρο pop-up στο οποίο μπορούμε να επιλέξουμε το εκτελέσιμο, επιλέγοντας τον φάκελο στο “Program” και διαλέγοντας το αρχείο.



Στα πλαίσια του εργαστηρίου, οι εργασίες θα πραγματοποιηθούν σε έτοιμα templates αποτελούμενα από τα απαραίτητα περιφερειακά. Για παράδειγμα, στην παρακάτω εικόνα εμφανίζεται το template που θα χρησιμοποιηθεί για προγραμματισμό και εμφάνιση αποτελεσμάτων σε γραφική οθόνη LCD 16x2.


AREF
100n
X1
XTAL1
XTAL2
8MHz
C1
22pF
C2
22pF
VCC
POWER
R50
10K
D2
1N4148
RESET
R51
220
C3
0.1u
DIPSW_8
DSW1
ATMEGA2560
LCD1
RXD
TXD
RTS
CTS
PE1
PE0
PF6
GND
RV0
VALUE=4
RV1
VALUE=4
PF0
PF1
1k
LED13
LED12
LED11
LED10
SW3
SW2
SW1
SW0
University of West Attica
Μικροελεγκτές & Ενσωματωμένα Συστήματα
ADC Evaluation Board
ATmega2560
Γρ. Καλτσάς

Ασκήσεις

1. Να γραφεί ένα πρόγραμμα το οποίο να ρυθμίζει την λειτουργία της πόρτας Α σαν είσοδο με ενεργοποιημένες τις pull-up αντιστάσεις και την λειτουργία των πορτών Β και D σαν εξόδους. Το πρόγραμμα να διαβάζει τις τιμές των 8 μικροδιακοπτών που είναι συνδεδεμένοι στην πόρτα Α και να τις εξάγει στις πόρτες Β και D με θετική και αρνητική λογική.

   1β. Τροποποιήστε το πρόγραμμα ώστε αν ο ακροδέκτης 0 της πόρτας Α είναι ενεργός (σε κατάσταση HI), να ενεργοποιούνται οι ακροδέκτες 0, 2, 4 και 6 της πόρτας Β, αλλιώς να ενεργοποιούνται οι ακροδέκτες 1, 3, 5 και 7 της ίδιας πόρτας.

2. Να δημιουργηθεί πρόγραμμα σε γλώσσα C για τον AVR το οποίο να υλοποιεί έναν απαριθμητής 0-10 στην πόρτα Β με συχνότητα 4Hz. Για να δημιουργήσετε μία χρονική καθυστέρηση χρησιμοποιήστε την συνάρτηση `delay_ms()`, η οποία δηλώνεται στο αρχείο `delay.h`:

```
void delay_ms(unsigned int n); /* Δημιουργείται καθυστέρηση n millisecs. */
```

3. Να δημιουργηθεί πρόγραμμα σε γλώσσα C για τον AVR το οποίο να ελέγχει το MSB και το LSB της πόρτας Α. Στην περίπτωση που αυτά είναι ίσα να προβάλλεται το περιεχόμενο της πόρτας Α στην πόρτα Β, ενώ σε αντίθετη περίπτωση να προβάλλεται το ανάστροφο περιεχόμενο της πόρτας Α στην πόρτα Ε.

4. Να δημιουργηθεί πρόγραμμα σε γλώσσα C για τον AVR το οποίο να ελέγχει την ισοτιμία της πόρτας C. Σε περίπτωση άρτιας ισοτιμίας να αναβοσβήνουν 10 φορές τα LEDS της πόρτας Β με συχνότητα 2Hz, ενώ σε αντίθετη περίπτωση να μένει αναμμένο το low nibble της πόρτας Β.

5. Να δημιουργηθεί πρόγραμμα σε γλώσσα C για τον AVR το οποίο να υλοποιεί έναν απαριθμητή (up counter) 0-15 στην πόρτα Β με συχνότητα 4 Hz και ταυτόχρονα έναν απαριθμητή (up counter) 0-3 στην πόρτα C με την ίδια συχνότητα.

6. Να γραφεί ένα πρόγραμμα το οποίο να ρυθμίζει την λειτουργία της πόρτας Α σαν έξοδο και την λειτουργία των πορτών Β και D σαν εισόδους με ενεργοποιημένες τις pull-up αντιστάσεις. Ο ακροδέκτης 0 της πόρτας Α να ενεργοποιείται μόνο αν οι ακροδέκτες 0 των πορτών Β και D είναι ενεργοί. Οι υπόλοιποι ακροδέκτες να ενεργοποιούνται αν ο ακροδέκτης 2 της πόρτας Β ή της πόρτας D είναι ενεργός, αλλιώς να απενεργοποιούνται.

7. Αναπτύξτε πρόγραμμα, το οποίο να αναβοσβήνει διαδοχικά το high nibble και το low nibble της θύρας Β με τυχαία συχνότητα. Τροποποιήσετε το πρόγραμμα αυτό, ώστε να αναβοσβήνουν τα περιττά (PB.1, PB.3, PB.5, PB.7) και τα άρτια (PB.0, PB.2, PB.4, PB.6) leds με τυχαία συχνότητα.

8. Αναπτύξτε πρόγραμμα το οποίο να θέτει την θύρα Β σαν έξοδο και να δημιουργεί σε αυτή έναν μετρητή Johnson (χρησιμοποιείστε τις εντολής bit shift left << και bit shift right >>). Στη συνέχεια τροποποιείστε το συγκεκριμένο πρόγραμμα ώστε το

ανακυκλούμενο 1 να μην διέρχεται από το C flag (μετρητής με διαδοχικές καταστάσεις 1, 2, 4, 8, 16, 32,64, 128)

9. Αναπτύξτε πρόγραμμα που δημιουργεί έναν μετρητή 16 bit στις PORTB και PORTC. Το low nibble του μετρητή να τρέχει στην PORTB και το high nibble στην PORTC.