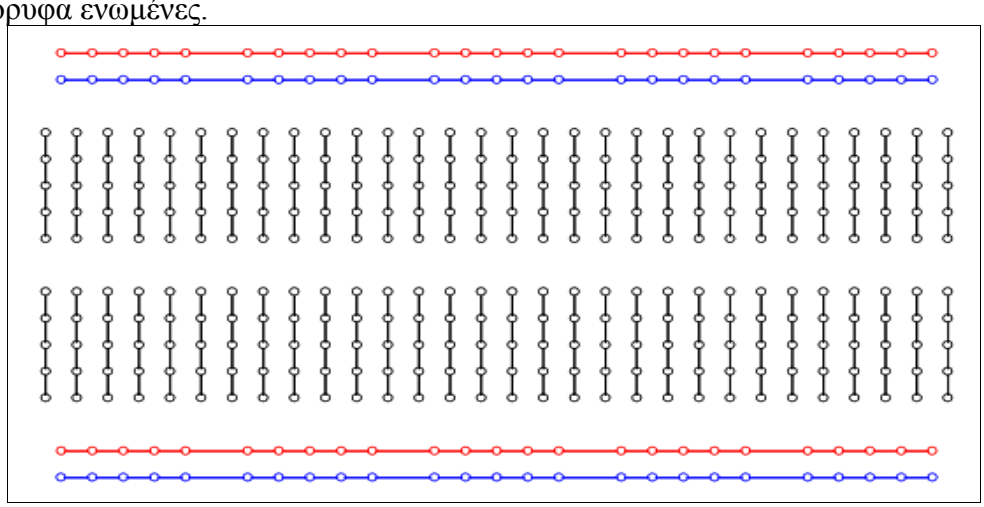
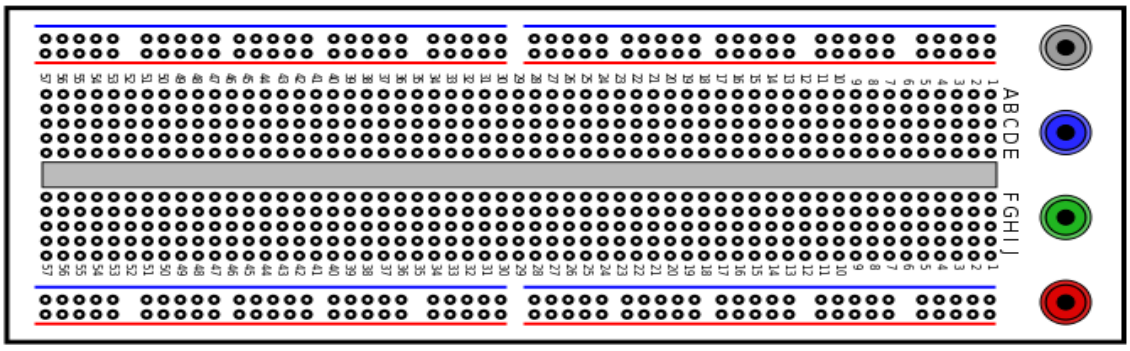
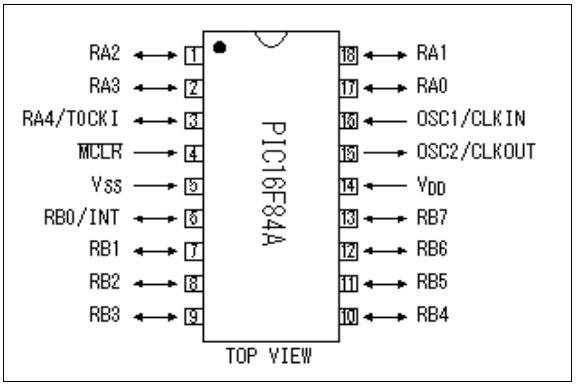
BreadBoard PIC

Στο άρθρο αυτό θα δούμε αναλυτικά την κατασκευή ενός συστήματος μικροϋπολογιστή κατασκευασμένο πάνω σε breadboard. Το σύστημα διαθέτει κοινούς μικροελεγκτές της οικογένειας PIC, οι οποίοι είναι κοινοί και φθηνοί. Το άρθρο απευθύνεται σε μαθητές λυκείου ή ΕΠΑΛ που έχουν γνώσεις προγραμματισμού άλλα δεν γνωρίζουν τον σχεδιασμό και την κατασκευή συστημάτων. Αρχικά δεν χρειάζονται εξειδικευμένες γνώσεις ηλεκτρονικών, αρκεί να τηρηθούν οι οδηγίες κατασκευής. Μετά ο καθένας μπορεί να εμβαθύνει στη τεχνολογία των εξαρτημάτων που χρησιμοποιεί.
Τι θα χρειαστούμε;

1. **Breadboard**. Η breadboard είναι μια πλακέτα διασύνδεσης εξαρτημάτων χωρίς να είναι απαραίτητη η κόλληση με κολλητήρι. Στο παρακάτω σχέδιο βλέπουμε την εσωτερική διασύνδεση μιας breadboard. Παρατηρείστε ότι οι δύο σειρές πάνω και οι δύο κάτω είναι οριζόντια ενωμένες και χρησιμοποιούνται για την διανομή των τάσεων τροφοδοσίας σ' όλο το μήκος της πλακέτας. Εσωτερικά έχουμε δύο πεντάδες από οπές ABCDE (πάνω) και FGHIJ (κάτω), οι οποίες είναι κατακόρυφα ενωμένες.





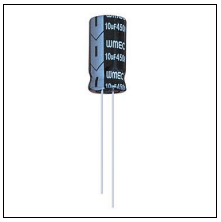


2. **Μικροελεγκτή PIC 16F84A**. Ο 16F84A διαθέτει μνήμη Flash ROM 2Kbytes, RAM 68 bytes και EEPROM 64 bytes. Μία πόρτα εισόδου-εξόδου των 8 bits και μία των 4 bits, συνολικά 12 bits. Μέγιστη συχνότητα λειτουργίας είναι τα 20MHz. Εναλλακτικά μπορούμε να χρησιμοποιήσουμε τον 16F628A ο οποίος είναι συμβατός στις θέσεις των pins αλλά έχει πολύ περισσότερες δυνατότητες.
3. **Κρύσταλλος 20MHz**. Ο κρύσταλλος συνδέεται στα ποδαράκια (pins) OSC1(15) και OSC2(16) και μαζί με δύο πυκνωτές πραγματοποιούν τον κρυσταλλικό ταλαντωτή. Ο ταλαντωτής παράγει τετραγωνικό παλμό συχνότητας 20.000.000 κύκλων, ο οποίος δίνει τους κατάλληλους χρονισμούς στα εσωτερικά κυκλώματα του PIC. Φυσικά μπορείτε να χρησιμοποιήσετε και κρύσταλλο μικρότερης συχνότητας αλλά θα πρέπει να το δηλώσετε στον μεταφραστή (compiler) όπως θα δούμε αργότερα.

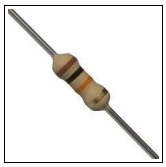
4. **Δύο πυκνωτές 22pF** κεραμικούς. Οι πυκνωτές πραγματοποιούν τον ταλαντωτή και συνδέουμε το ένα ποδαράκι στο 15 του PIC και το άλλο στη γη GND. Τον άλλο στο 16 του PIC και στη γη. Οι συγκεκριμένοι δεν έχουν πολικότητα και τους βάζουμε όπως θέλουμε. Η τάση λειτουργίας μπορεί να είναι 60V ή και μικρότερη.

5. **Ένα πυκνωτή 100nF** πλαστικό ή multilayer ή κεραμικό. Ο πυκνωτής αυτός πρέπει να γράφει 104 ή 100nF ή 0.1uF. Η τάση λειτουργίας 50V και δεν έχει πολικότητα. Η χρήση του είναι η απόζευξη και συνδέεται με το pin 14 του PIC όπου πάει η τάση τροφοδοσίας των 5 – 5,5V και τη γη. Διάφορες παρασιτικές εναλλασσόμενες τάσεις που εμφανίζονται μαζί με την συνεχή τάση τροφοδοσίας, γειώνονται μέσω του πυκνωτή ώστε στο pin 14 να υπάρχει καθαρή συνεχής τάση.

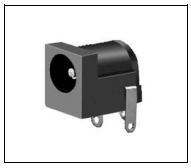
6. **Ένα πυκνωτή 1μF** ηλεκτρολυτικό. Οι ηλεκτρολυτικοί πυκνωτές έχουν πολικότητα και το μακρύ ποδαράκι είναι το +. Επίσης υπάρχει και η ένδειξη (-) μείον πάνω στον πυκνωτή. Η τάση λειτουργίας 16V. Αυτός ο πυκνωτής συνδέεται στο pin 4 του PIC (εκεί πάει το +), το (-) πάει στη γείωση. Μαζί με μια αντίσταση 10K πραγματοποιούν το κύκλωμα του reset.



7. **Μία αντίσταση 10K**. Η αντίσταση έχει χρώματα καφέ – μαύρο – πορτοκαλί και πρέπει να είναι ισχύος 1/4W. Οι αντιστάσεις γενικώς δεν έχουν πολικότητα. Το ένα ποδαράκι συνδέεται στο pin 14 μαζί με το (+) του πυκνωτή 1μF και το άλλο στα +5V.

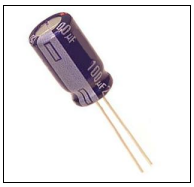


8. **Ένα DC jack**. Εδώ συνδέεται το τροφοδοτικό 6V. Αυτό έχει 3 ποδαράκια και εμείς συνδέουμε τα 2. Το πίσω είναι το (+) και το μπροστινό είναι το 0V ή γη ή GND. Το πλαϊνό δεν το χρησιμοποιούμε. Εναλλακτικά μπορούμε να μη βάλουμε αυτό το εξάρτημα, αν δώσουμε τάση από τη θύρα USB του υπολογιστή μας. Επειδή η θύρα USB βγάζει ακριβώς 5V δεν θα χρειαστούμε και την δίοδο 1N4004.



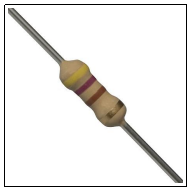
9. **Μια δίοδο 1N4001** ή 1N4004. Η δίοδος είναι μια επαφή PN και επιτρέπει την ροή του ρεύματος μόνο προς τη μία κατεύθυνση. Από το ένα ποδαράκι υπάρχει ένα άσπρο ή ασημί δαχτυλίδι. Αυτή είναι η κάθοδος και πάει στην πλευρά όπου θέλουμε να πάει το ρεύμα. Εμείς συνδέουμε την κάθοδο με το (+) της πλακέτας και την άνοδο με το πίσω ποδαράκι του dc jack. Εδώ η δίοδος κάνει διπλή δουλειά, πρώτα προστατεύει το κύκλωμα από λάθος σύνδεση της πολικότητας του τροφοδοτικού και μετά κάνει μια πτώση τάσης (0,5 – 0,6V) ώστε τα 6V του τροφοδοτικού να γίνουν 5,4 – 5,5V που είναι κατάλληλα για τον PIC.

10. **Ένας πυκνωτής 100μF** στα 16V ηλεκτρολυτικός. Αυτός κάνει εξομάλυνση της τάσης του τροφοδοτικού ώστε στο (+) της πλακέτας να έχουμε καθαρή συνεχή τάση +5 ή +5,5V. Έχει πολικότητα και το (+) συνδέεται στο (+) της πλακέτας και το (-) στο GND.

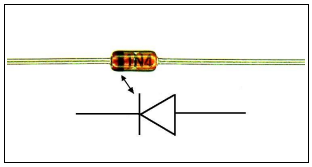


11. **Τρία Led κόκκινα**. Τα Led είναι δίοδοι που εκπέμπουν φως. Έχουν πολικότητα και το μεγάλο ποδαράκι είναι το (+). Αν μπουν ανάποδα απλά δεν ανάβουν. Εμείς τα χρησιμοποιούμε ως συσκευές εξόδου ώστε να βλέπουμε την κατάσταση του μικροελεγκτή μας. Οι κάθοδοι πάνε στο GND ενώ οι άνοδοι (+) πάνε στα bit 1, 2, 3 της πόρτας 'B' μέσω τριών αντιστάσεων.

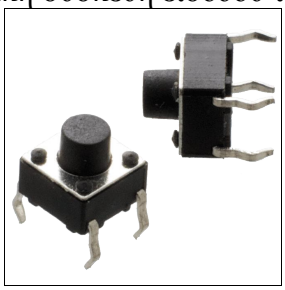
12. **Τέσσερις αντιστάσεις 470 Ohm**. Έχουν χρώματα κίτρινο – μοβ – καφέ και είναι ισχύος 1/4Watt. Οι τρεις πάνε σε σειρά με τα Led και η χρήση τους είναι να μειώσουν την ένταση του ρεύματος που ρέει στα led ώστε να μην καταστραφούν. Η τέταρτη συνδέεται σε σειρά με το push button και μπορεί να αντικατασταθεί με απλό σύρμα.



13. **Μια δίοδος 1N4148**. Η άνοδος αυτής συνδέεται με το προγραμματιστή ICSP και η κάθοδος (μαύρο δαχτυλίδι) πάει στο master clear -MCLR του PIC.

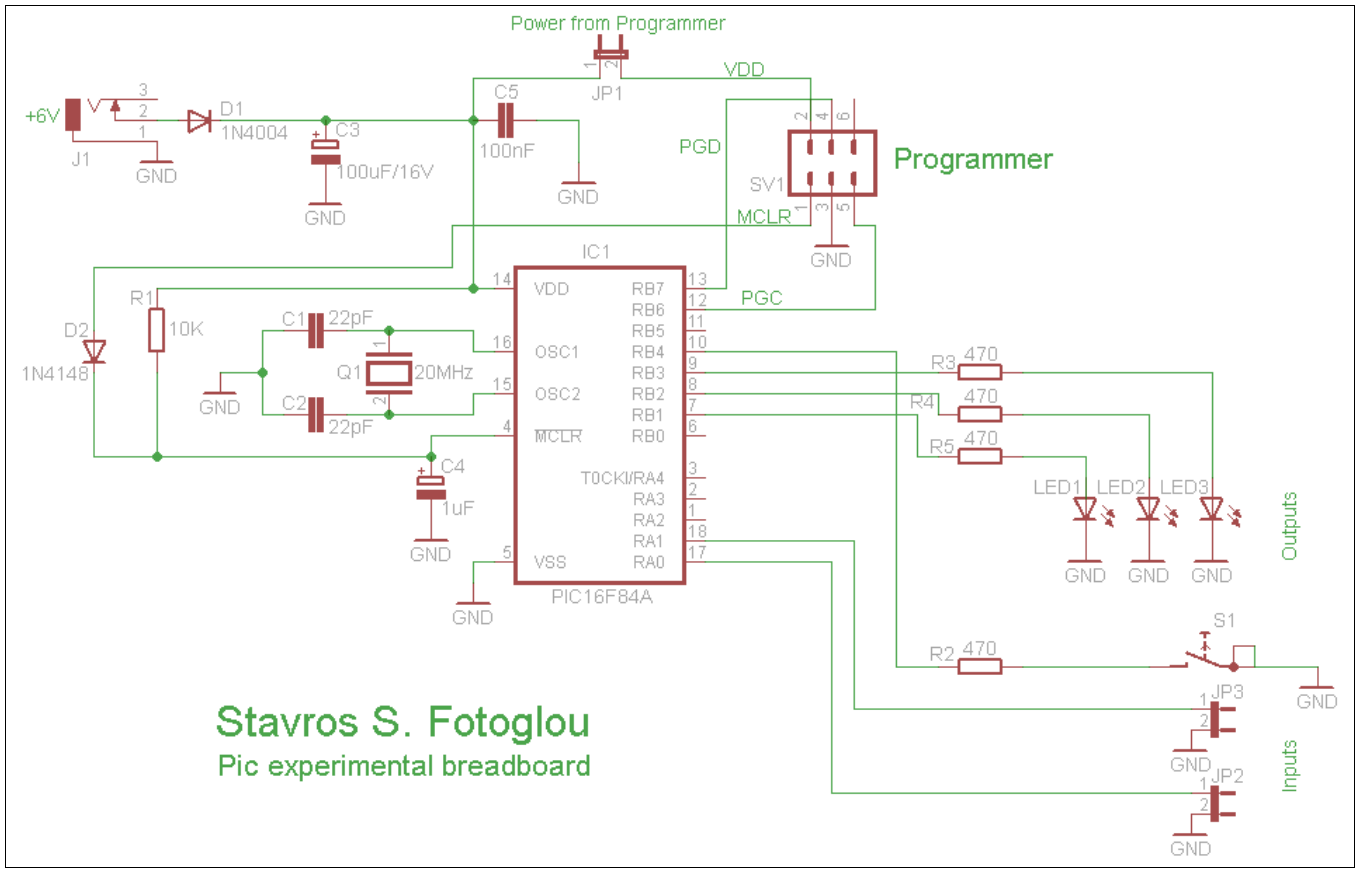


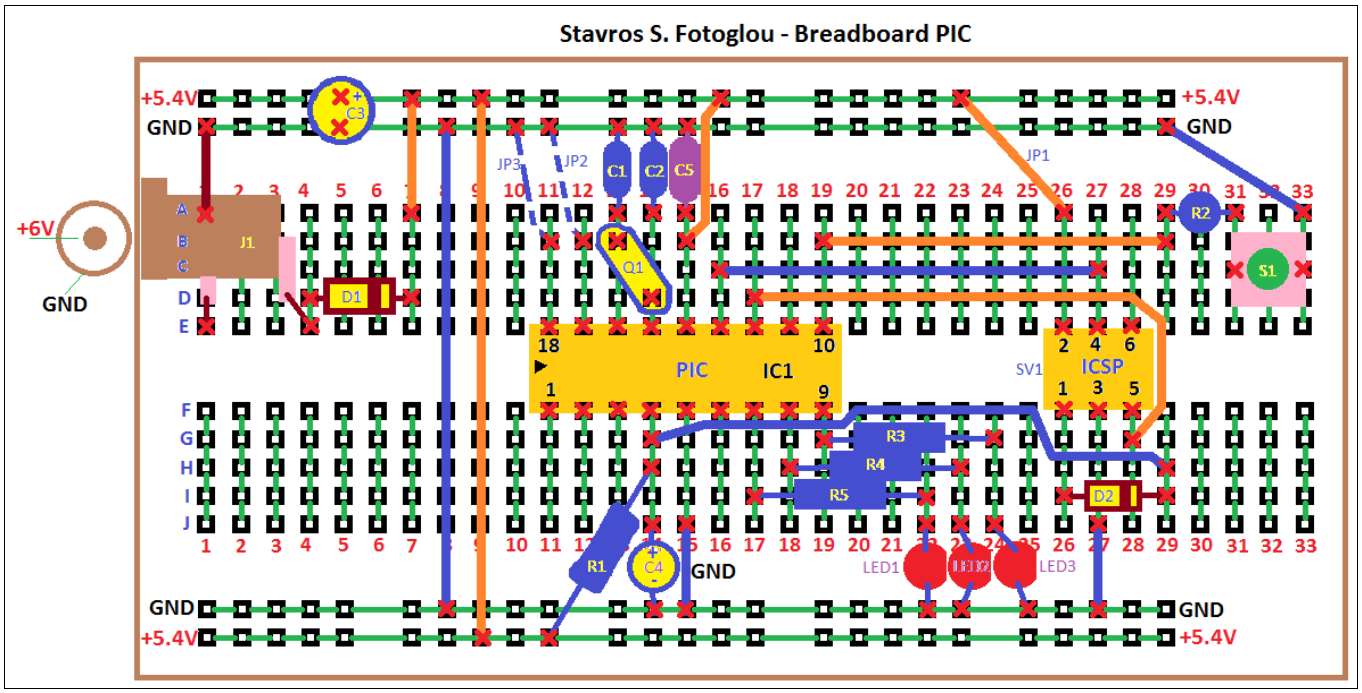
14. **Ένα Push Button**. Είναι η μοναδική συσκευή εισόδου της κατασκευής μας.



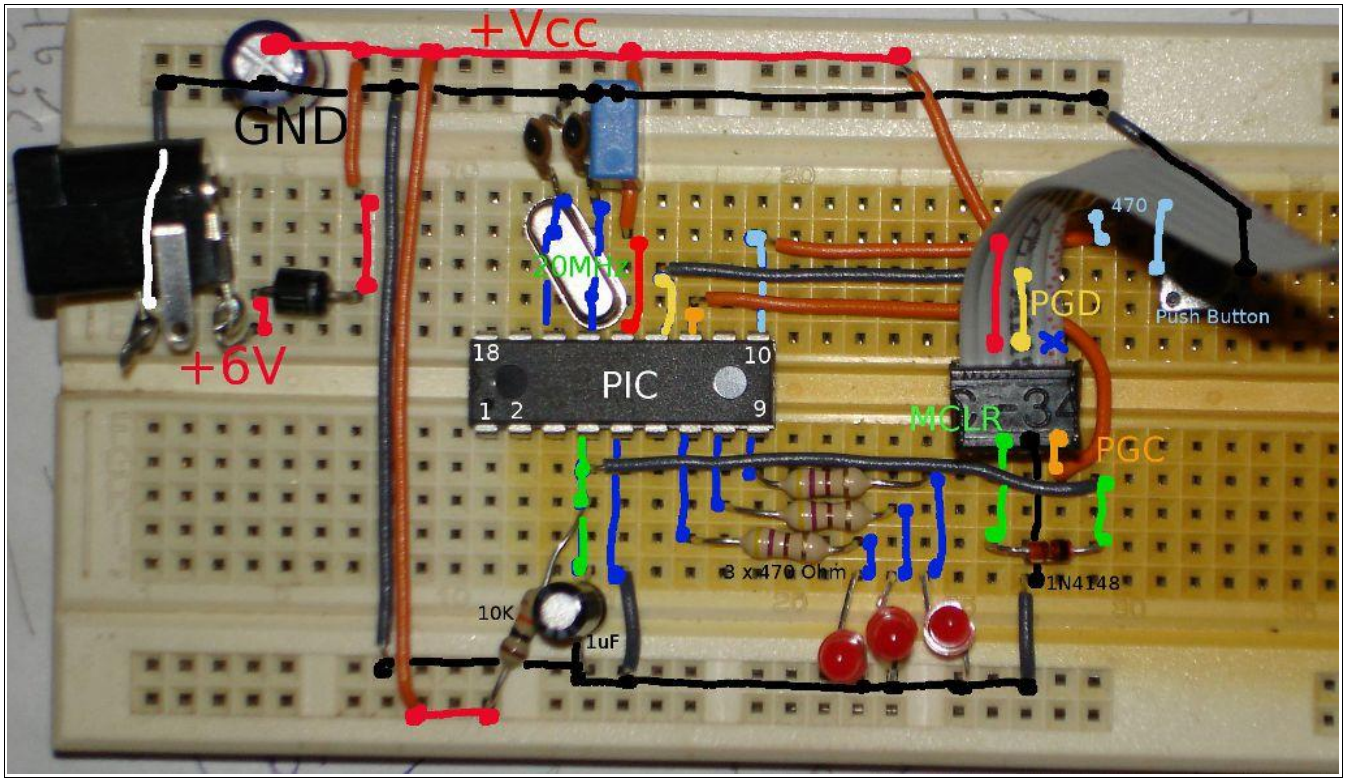
15. **Καλώδια μονόκλωνα** για τις συνδέσεις.

Ακολουθεί το θεωρητικό κύκλωμα της κατασκευής μας:

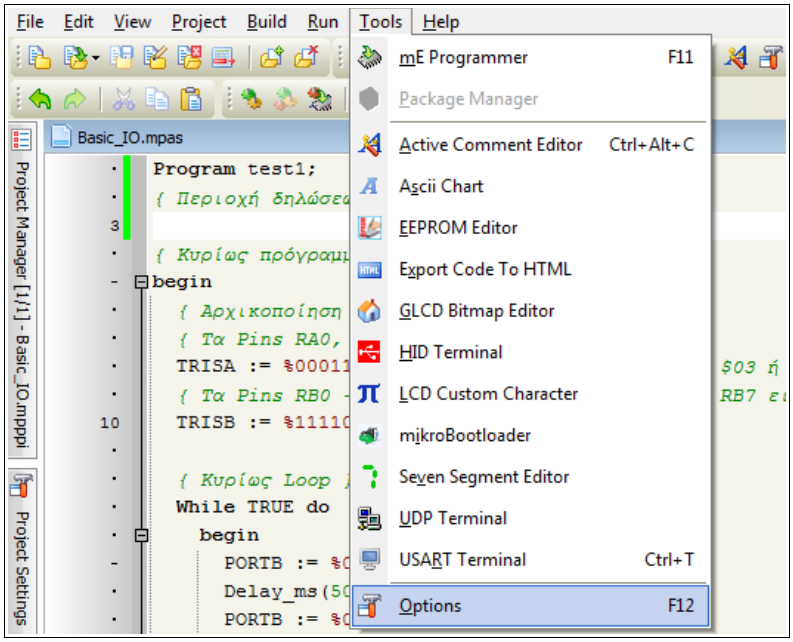




Στη συνέχεια βλέπετε μια υλοποίηση της κατασκευής πάνω σε μια Breadboard. Η συγκεκριμένη έχει 64 στήλες αλλά εμείς χρησιμοποιούμε μόνο τις 33. Τα jumpers JP2, JP3 δεν χρειάζεται να τα βάλετε αρχικά. Στη συνέχεια βλέπετε την κατασκευή όπως είναι στη πραγματικότητα.

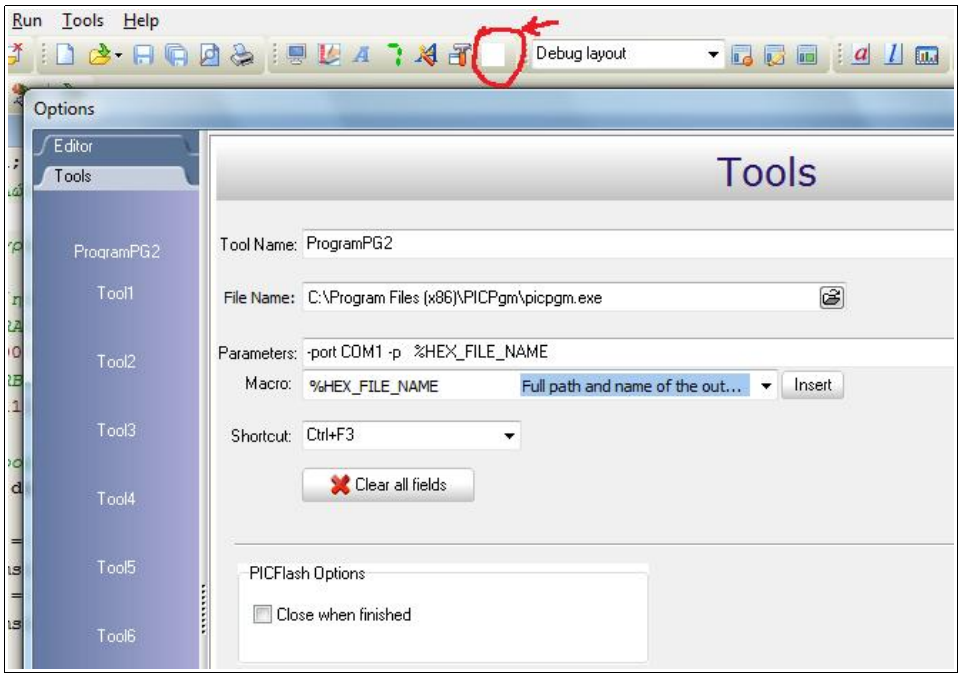


Για τον προγραμματισμό του μικροελεγκτή θα χρησιμοποιήσουμε τον προγραμματιστή PIC-PG2 της OLIMEX που έχω δημοσιεύσει στο http://users.sch.gr/stfotoglou/joomla/index.php?option=com_content&task=view&id=92&Itemid=1 , και για λογισμικό το PICPgm programmer. Αρχικά θα γράψουμε το πρώτο μας πρόγραμμα σε Pascal και συγκεκριμένα με το IDE MikroPascal for PIC της εταιρίας Mikroelektronika. Υπάρχει δωρεάν πλήρως λειτουργική έκδοση με τον περιορισμό ότι το μέγεθος του προγράμματος δεν ξεπερνάει τα 2Kbytes. Για τις δικές μας ανάγκες είναι υπέρ αρκετή.
Πρέπει να ρυθμίσουμε ένα εργαλείο το οποίο θα χρησιμοποιεί τον PICPgm programmer. Πάμε Tools – Options.



Μετά εντοπίζουμε το Tool 0 και δίνουμε ένα νέο όνομα π.χ. PrpgramPG2. Εντοπίζουμε το απόλυτο μονοπάτι του εκτελέσιμου που το βάζουμε στο File Name και συμπληρώνουμε το πεδίο Parameter: **`-port COM1 -p %HEX_FILE_NAME`**. Στο port βάζουμε την σειριακή θύρα που έχουμε συνδέσει τον programmer. Αν θέλουμε να γράψουμε μόνο την μνήμη flash, αντί για **`-p %HEX_FILE_NAME`** γράφουμε

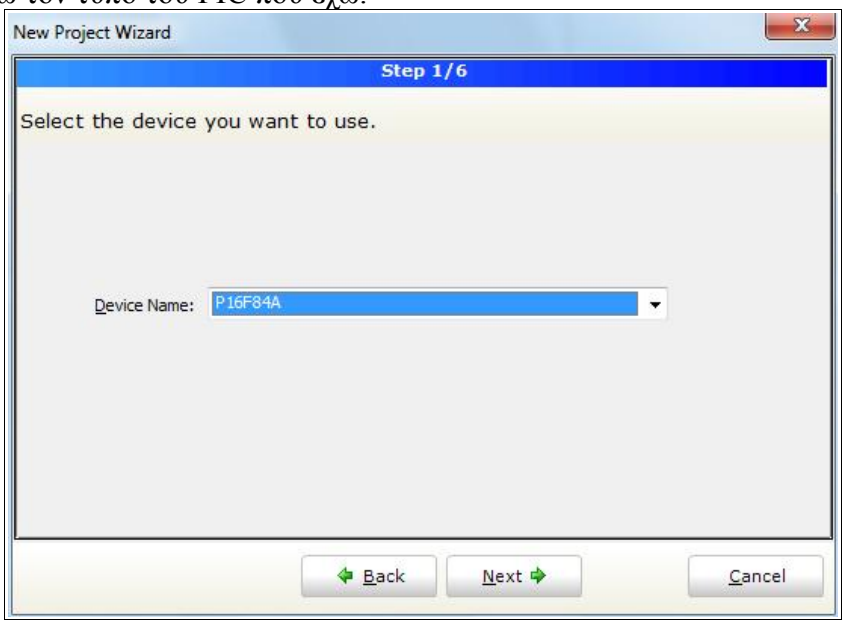
`-p_code %HEX_FILE_NAME` .
Αν θέλουμε δίνουμε και μια συντόμευση πλήκτρων.



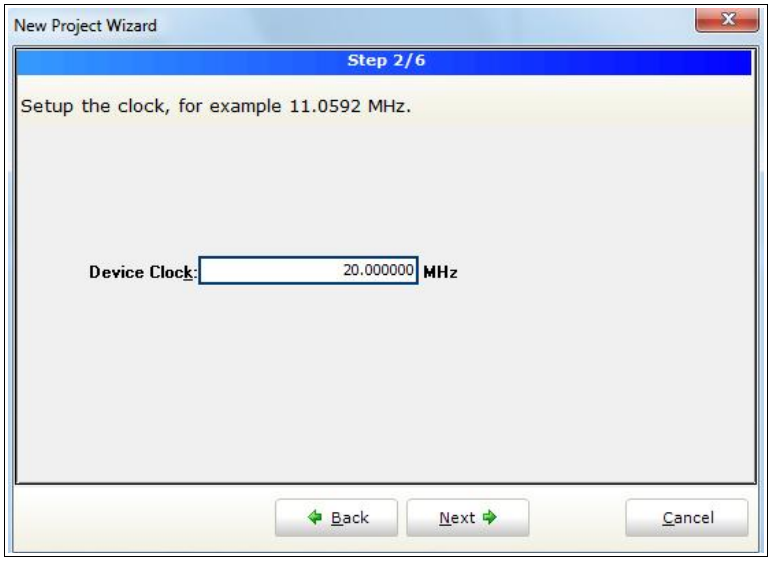
Όταν πατήσουμε Apply, πάνω στην γραμμή εργαλείων εμφανίζεται ένα εργαλείο άσπρο χωρίς εικονίδιο το οποίο θα πατάμε για να προγραμματίσουμε τον PIC. Εναλλακτικά για το παράδειγμά μας πατάμε και CTRL + F3 που είναι η συντόμευση.

Το πρώτο μας πρόγραμμα

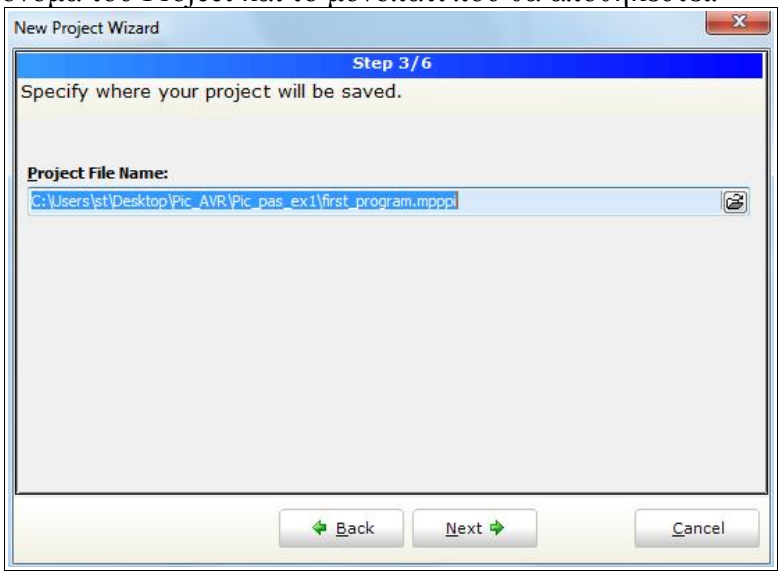
Πατάμε File – New – New Project και ανοίγει ένα wizard ως εξής:
Στο 1ο βήμα επιλέγω τον τύπο του PIC που έχω.



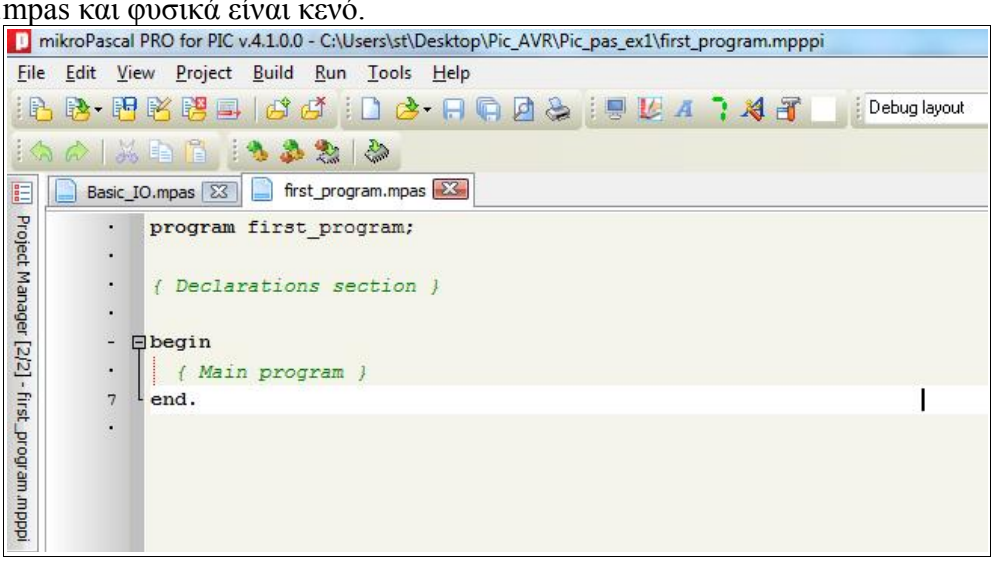
Στο 2ο βήμα βάζω την συχνότητα χρονισμού του PIC.



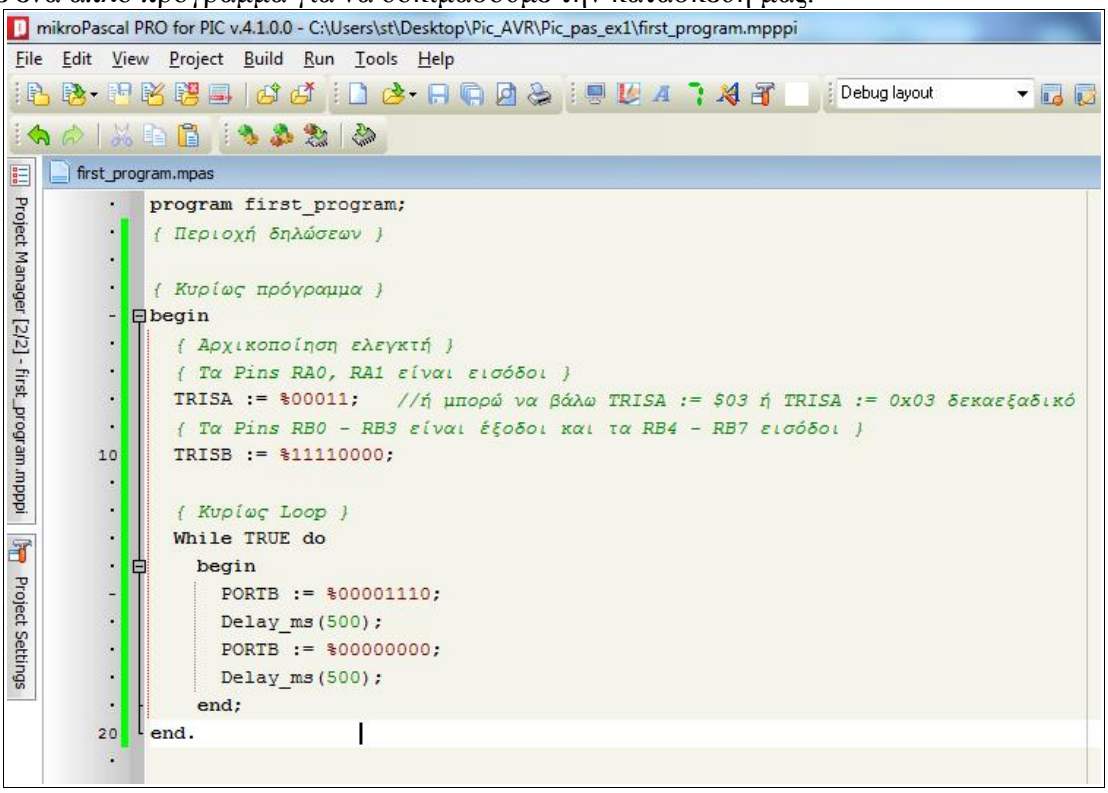
Στο 3ο βήμα ορίζω το όνομα του Project και το μονοπάτι που θα αποθηκευτεί.



Στο 4ο βήμα προσθέτουμε αρχεία στο project. Εδώ δεν βάζουμε κανένα αρχείο γιατί θα το γράψουμε στη συνέχεια.
Στο 5ο βήμα ρωτάει αν θα εισάγουμε όλες τις έτοιμες βιβλιοθήκες και απαντάμε καταφατικά.
Στο 6ο βήμα έχουμε τελειώσει και ανοίγει ο editor με ένα αρχείο το οποίο ονομάζεται first_program.mpas και φυσικά είναι κενό.



Γράφουμε ένα απλό πρόγραμμα για να δοκιμάσουμε την κατασκευή μας.



Εδώ είναι ο κώδικας σε pascal. Στη γραμμή 8 ορίζω ποια ποδαράκια της πόρτας Α θα είναι έξοδοι και ποια είσοδοι. Στη γραμμή 10 γίνεται το ίδιο για την πόρτα Β. Από την 13 έως την 19 δημιουργώ ένα ατέρμονο loop. Αυτό είναι το κύριο loop που εκτελείτε συνέχεια όσο ο PIC είναι σε λειτουργία. Στη γραμμή 15 βγάζω στη πόρτα Β τον αριθμό 00001110. Στη πραγματικότητα εκεί που είναι άσοι είναι τα 3 Led, τα οποία θα ανάψουν. Μετά ακολουθεί μια καθυστέρηση μισού δευτερολέπτου και μετά βγάζω στη πόρτα 00000000 και σβήνω τα 3 Led. Περιμένω μισό δευτερόλεπτο και εκτελώ πάλι την γραμμή 15. Δηλαδή αυτό το πρόγραμμα αναβοσβήνει τα τρία Led με περίοδο ενός δευτερολέπτου.

```
1. program first_program;
2. { Περιοχή δηλώσεων }
3.
4. { Κυρίως πρόγραμμα }
5. begin
6.   { Αρχικοποίηση ελεγκτή }
7.   { Τα Pins RA0, RA1 είναι εισόδοι }
8.   TRISA := %00011;   //ή μπορώ να βάλω TRISA := $03 ή TRISA := 0x03 δεκαεξαδικό
9.   { Τα Pins RB0 - RB3 είναι έξοδοι και τα RB4 - RB7 εισόδοι }
10.       TRISB := %11110000;
11.
12.       { Κυρίως Loop }
13.       While TRUE do
14.         begin
15.           PORTB := %00001110;
16.           Delay_ms(500);
17.           PORTB := %00000000;
18.           Delay_ms(500);
19.         end;
20.     End.
```

Ύστερα πατάμε Build και αν δεν έχουμε συντακτικά λάθη, παράγεται το αποτέλεσμα σε γλώσσα μηχανής PIC.

Messages | Quick Converter

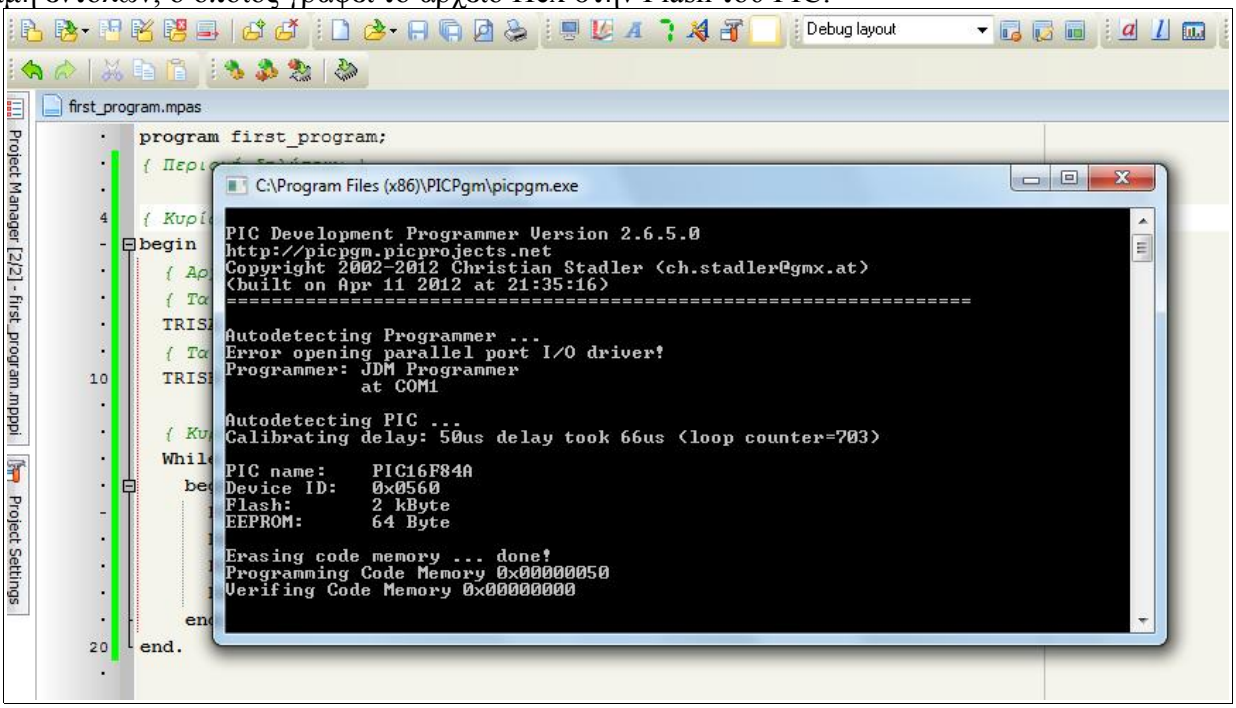
Errors | Warnings | Hints

Line	Message No.	Message Text	Unit
20	1010	Hint: Unit "first_program.mpas" has been recompiled	first_program.mpas
0	134	Compiled Successfully	C:\Users\st\Desktop\Pic_AVR\Pic_pas_ex1\first_program...
0	139	All files Compiled in 16 ms	
0	1144	Used RAM (bytes): 2 (4%) Free RAM (bytes): 50 (96%)	Used RAM (bytes): 2 (4%) Free RAM (bytes): 50 (96%)
0	1144	Used ROM (program words): 37 (4%) Free ROM (program words): 987 (96%)	Used ROM (program words): 37 (4%) Free ROM (progra...
0	145	Project Linked Successfully	first_program.mpppi
0	140	Linked in 62 ms	
0	141	Project 'first_program.mpppi' completed: 140 ms	
0	103	Finished successfully: 27 Μαρ 2013, 00:36:28	first_program.mpppi

Παρατηρούμε ότι το πρόγραμμα καταλαμβάνει μόλις 37 words της μνήμης Flash.

Εγγραφή στον PIC

Πατάμε το άσπρο κενό εικονίδιο που προγραμματίσαμε πριν ή Ctrl + F3 και ανοίγει ο PICPgm σε γραμμή εντολών, ο οποίος γράφει το αρχείο Hex στην Flash του PIC.



Η ίδια εταιρία διαθέτει και τον compiler MikroBasic for Pic. Να το ίδιο πρόγραμμα γραμμένο σε basic:

```
1.program first_program
2.' Περιοχή δηλώσεων
3.
4.'Κυρίως πρόγραμμα
5.main:
6.'   Αρχικοποίηση ελεγκτή
7.    TRISA = 0x03    '$03
8.    TRISB = %11110000
9.'  Κυρίως Loop
10.         While TRUE
11.           PORTB = %00001110
12.           Delay_ms(500)
13.           PORTB = %00000000
14.           Delay_ms(500)
15.         Wend
16.     End.
```

Τέλος το ίδιο θα το γράψουμε με τον compiler MikroC for PIC.

```
1.void main()
2.{
3.  TRISA = 03;
```

```
4.  TRISB = 0b11110000;
5.  while (1)
6.    {
7.     PORTB = 0b00001110;
8.     delay_ms(500);
9.     PORTB = 0b00000000;
10.          delay_ms(500);
11.        }
12.     }
```

Εντολή εισόδου - έλεγχος του Button

Στη λίστα που ακολουθεί ελέγχουμε αν πατήθηκε το Button και ανάβουμε το Led1. Για να είναι το πρόγραμμα πιο δομημένο δηλώνουμε με const (σταθερές) τα ποδαράκια της πόρτας και την αντίστοιχη λειτουργία π.χ. LED1 κλπ.

Εκτελώντας την εντολή στη γραμμή 18 ενεργοποιούμε την αντίσταση pull-up για τις εισόδους της πόρτας B. Η pull-up είναι μια αντίσταση εσωτερικά στον pic που συνδέεται στο +Vdd. Δηλαδή το RB4 είναι στα 5V και αν το button δεν είναι πατημένο διαβάζω μόνιμα λογικό '1' . Όταν πατηθεί το button θα διαβάζω λογικό '0' γιατί θα περάσει ρεύμα από το button προς το GND και η τάση θα γίνει 0V. Αν δεν βάλω την εντολή της γραμμής 18 θα πρέπει να συνδέσω μια αντίσταση 10K (μαύρο – κόκκινο – πορτοκαλί) από το RB4 στο +5V.

Η εντολή στη γραμμή 25 και 28 θα μπορούσε να αντικατασταθεί με **`if PORTB and %00010000 = 0 then`** , αλλά είναι πιο δομημένο το πρόγραμμα με την χρήση των const. Και στις δύο περιπτώσεις ο compiler θα δώσει το ίδιο μέγεθος στο εκτελέσιμο αρχείο hex.

```
1. program second_program;
2. { Περιοχή δηλώσεων }
3.  const RBPU = 7;   //To bit 7 του OPTION Register
4.  const LED1 = 1;   //Led συνδεδεμένο στο RB1
5.  const LED2 = 2;   //Led συνδεδεμένο στο RB2
6.  const LED3 = 3;   //Led συνδεδεμένο στο RB3
7.  const BUTTON1 = 4;      //Push Button στο RB4
8.  const DEBOUNCE_TIME = 25;      //Χρόνος debounce σε msec
9.
10. { Κυρίως πρόγραμμα }
11. begin
12.   { Αρχικοποίηση ελεγκτή}
13.   { Τα Pins RA0, RA1 είναι είσοδοι}
14.   TRISA := %00011;
15.   { Τα Pins RB0 - RB3 είναι έξοδοι και τα RB4 - RB7 είσοδοι }
16.   TRISB := %11110000;
17.   (* Ενεργοποίηση της αντίστασης PULL UP για την πόρτα B*)
18.   OPTION_REG := OPTION_REG And Not(1 Shl RBPU);
19.   (* Σβήσε όλα τα Led *)
20.   PORTB := 0;
21.
22.   (*** Κυρίως LOOP ***)
23.   While TRUE do
24.     begin
25.       if PORTB and (1 Shl BUTTON1) = 0 then //Ολίσθηση αριστερά 4 θέσεις το 1 δηλ.
26.          begin                              //00010000 και αν με AND δώσει 0 τότε πατημένο
27.            delay_ms(DEBOUNCE_TIME);         //Περίμενε λίγο να σταθεροποιηθεί η επαφή
28.            if PORTB and (1 Shl BUTTON1) = 0 then       //Συνεχίζει ;
29.               PORTB := PORTB or (1 shl LED1);          //Βγάλε στη πόρτα xxxxxx1x δηλ.
30.          end                                           //άναψε το Led1
31.       else                                  //Αλλιώς
32.          PORTB := PORTB and not(1 shl Led1);       //Σβήσε το Led1
33.     end;
34. End.
```

Στη συνέχεια ακολουθεί το ίδιο πρόγραμμα αλλά κάνει χρήση διαδικασίας για το άναμμα του Led και συνάρτησης για τον έλεγχο πατήματος του κουμπιού. Ο μεταφρασμένος κώδικας έχει σχεδόν τριπλασιαστεί (ήταν 37 bytes και έγινε 113), αλλά το πρόγραμμα είναι πολύ πιο δομημένο. Φυσικά αν χρειαστούμε περισσότερες κλήσεις των συναρτήσεων τότε το νέο πρόγραμμα θα είναι πολύ πιο μικρό από το μονολιθικό προηγούμενο.

```
1. program third_program;
2. { Περιοχή δηλώσεων }
3.  const RBPU = 7;   //To bit 7 του OPTION Register
4.  const LED1 = 1;   //Led συνδεδεμένο στο RB1
```

```
5.  const LED2 = 2;   //Led συνδεδεμένο στο RB2
6.  const LED3 = 3;   //Led συνδεδεμένο στο RB3
7.  const BUTTON1 = 4;      //Push Button στο RB4
8.  const DEBOUNCE_TIME = 25;      //Χρόνος debounce σε msec
9.
10.  (***          Διαδικασία η οποία ανάβει ή σβήνει κάποιο LED          ***
11.   *     Δέχεται ως παραμέτρους το bit της πόρτας B που είναι το LED     *
12.   ***  και True για να ανάψει ή False για να σβήσει.                   ***)
13.  Procedure SetLed(LedNo : Byte; State : Boolean);
14.   Begin
15.     if State = True then               //Αν είναι True
16.        PORTB := PORTB or (1 shl LedNo)  //Κάνε OR με '1' το bit του Led
17.     else                               //Αλλιώς
18.        PORTB := PORTB and not(1 shl LedNo);     //Κάνε AND με '0' το bit του Led
19.   End;
20.
21.  (***        Συνάρτηση η οποία ελέγχει αν πατήθηκε ένα κουμπί                ***
22.   *   Δέχεται ως παράμετρο το bit της πόρτας B που είναι συνδεδεμένο το κουμπί  *
23.   *** Επιστρέφει True αν είναι πατημένο, διαφορετικά False                    ***)
24.   Function ButtonIsPressd (ButtonId : Byte) : Boolean;
25.    Begin
26.      ButtonIsPressd := False;               //Αρχικά false
27.      if PORTB and (1 Shl ButtonId)  = 0 then //Ολίσθηση αριστερά x θέσεις το 1
28.         begin                               //και αν με AND δώσει 0 τότε πατημένο
29.             delay_ms(DEBOUNCE_TIME);        //Περίμενε λίγο να σταθεροποιηθεί η επαφή
30.             if PORTB and (1 Shl ButtonId) = 0 then
31.                ButtonIsPressd := True;      //Αν είναι ακόμη πατημένο επέστρεψε true
32.         end;
33.    End;
34.
35.  { Κυρίως πρόγραμμα }
36.  begin
37.    { Αρχικοποίηση ελεγκτή}
38.    { Τα Pins RA0, RA1 είναι είσοδοι}
39.    TRISA := %00011;
40.    { Τα Pins RB0 - RB3 είναι έξοδοι και τα RB4 - RB7 είσοδοι }
41.    TRISB := %11110000;
42.    (* Ενεργοποίηση της αντίστασης PULL UP για την πόρτα B*)
43.    OPTION_REG := OPTION_REG And Not(1 Shl RBPU);
44.    (* Σβήσε όλα τα Led *)
45.    PORTB := 0;
46.
47.    (*** Κυρίως LOOP ***)
48.    While TRUE do
49.      begin
50.        if ButtonIsPressd(BUTTON1) = True then   //Αν το κουμπί είναι πατημένο
51.           SetLed(Led1, True)                    //Άναψε το Led1
52.        else                                     //Αλλιώς
53.           SetLed(Led1, False);                  //Σβήσε το Led1
54.      end;
55.  End.
```

Άσκηση 1

Να τροποποιηθεί το παραπάνω πρόγραμμα ώστε να φτιάξουμε ένα δυαδικό απαριθμητή των τριών bits. Κάθε φορά που θα πατάμε το κουμπί να αυξάνει η αρίθμηση από 0 (000 όλα σβηστά) έως 7 (111 όλα αναμμένα). Όταν είναι στο 7 και πατήσουμε το κουμπί θα γυρίσει σε 0.

```
1. Program askisi1;
2. { Περιοχή δηλώσεων }
3.  const RBPU = 7;   //Το bit 7 του OPTION Register
4.  const LED1 = 1;   //Led συνδεδεμένο στο RB1
5.  const LED2 = 2;   //Led συνδεδεμένο στο RB2
6.  const LED3 = 3;   //Led συνδεδεμένο στο RB3
7.  const BUTTON1 = 4;      //Push Button στο RB4
8.  const DEBOUNCE_TIME = 25;      //Χρόνος debounce σε msec
9.
10.   var i, tmp : byte;
11.       blockFlag : boolean;
12.
13.  (***        Συνάρτηση η οποία ελέγχει αν πατήθηκε ένα κουμπί                ***
14.   *   Δέχεται ως παράμετρο το bit της πόρτας B που είναι συνδεδεμένο το κουμπί  *
15.   *** Επιστρέφει True αν είναι πατημένο, διαφορετικά False                    ***)
16.   Function ButtonIsPressd (ButtonId : Byte) : Boolean;
17.    Begin
18.      ButtonIsPressd := False;               //Αρχικά false
19.      if PORTB and (1 Shl ButtonId)  = 0 then //Ολίσθηση αριστερά x θέσεις το 1
20.         begin                               //και αν με AND δώσει 0 τότε πατημένο
21.             delay_ms(DEBOUNCE_TIME);        //Περίμενε λίγο να σταθεροποιηθεί η επαφή
```

```
22.            if PORTB and (1 Shl ButtonId) = 0 then
23.               ButtonIsPressd := True;      //Αν είναι ακόμη πατημένο επέστρεψε true
24.         end;
25.    End;
26.
27.  { Κυρίως πρόγραμμα }
28.  begin
29.    { Αρχικοποίηση ελεγκτή}
30.    { Τα Pins RA0, RA1 είναι είσοδοι}
31.    TRISA := %00011;
32.    { Τα Pins RB0 - RB3 είναι έξοδοι και τα RB4 - RB7 είσοδοι }
33.    TRISB := %11110000;
34.    (* Ενεργοποίηση της αντίστασης PULL UP για την πόρτα B*)
35.    OPTION_REG := OPTION_REG And Not(1 Shl RBPU);
36.    (* Σβήσε όλα τα Led *)
37.    PORTB := 0;
38.    i := 0;
39.    blockFlag := False;
40.    (*** Κυρίως LOOP ***)
41.    While TRUE do
42.      begin
43.        if (ButtonIsPressd(BUTTON1) = True) And (Not blockFlag) then   //Αν το κουμπί είναι πατημένο
44.           begin
45.             blockFlag := True;       //Μπλοκάρισε ώστε να εκτελεστούν μια φορά αυτές οι εντολες
46.             if i < 7 then            //Αν το i μικρότερο από 7
47.                i := i + 1            //Αυξησέ το κατά 1
48.             else                     //Αλλιώς
49.                i := 0;               //Μηδενισέ το
50.             // Ολίσθησε αριστερά το i μια θέση (αν ήταν 00000101 να γίνει 00001010) και βάλτο στην
51.             // μεταβλητή tmp. Αυτό το κάνουμε γιατί τα Led ξεκινούν στο bit1 και όχι
52.             // στο bit0 της πόρτας B
53.             tmp := i Shl 1;
54.             // Κάνε AND mask μόνο τα 3 bit που αφορούν τα Led
55.             tmp := tmp And %00001110;
56.             // Βγάλε τον μετρητή στα Led χωρίς να επηρεάσεις τα άλλα bits της πόρτας B
57.             PORTB := (PORTB And %11110001) Or tmp;
58.           end
59.        //Αλλιώς αν το κουμπί έχει αφεθεί τότε
60.        else if ButtonIsPressd(BUTTON1) = False then
61.           blockFlag := False;       //Ξεμπλοκάρισε για να επιτραπεί το επόμενο πάτημα
62.      delay_ms(5);                   //Καθυστέρηση 5 ms
63.      end;
64.  End.
```

Ακολουθεί το ίδιο πρόγραμμα γραμμένο σε 'C'.

```
1. //Ορισμοί σημάτων και σταθερών
2. #define RBPU            7
3. #define LED1            1
4. #define LED2            2
5. #define LED3            3
6. #define BUTTON1         4
7. #define DEBOUNCE_TIME   25
8. #define TRUE            1
9. #define FALSE           0
10.
11. typedef unsigned short u08;   //Ορισμός τύπου u08
12.
13. u08 Button_is_pressd(u08);    //Ορισμός πρωτοτύπου συνάρτησης
14.
15. //-------- Καθολικές μεταβλητές ----------------------
16. u08 i, tmp, blockFlag;
17.
18. //-------- Κυρίως πρόγραμμα --------------------------
19. void main()
20. {
21.  //Αρχικοποίηση PIC
22.  TRISA = 0b11100;
23.  TRISB = 0b11110000;
24.  OPTION_REG &= ~(1 << RBPU);
25.  PORTB = 0;
26.  i = 0;
27.  //Κυρίως LOOP
28.  while (1)
29.    {
30.     if ((Button_is_pressd(BUTTON1)) && (!blockFlag))
31.        {
32.         blockFlag = TRUE;
33.         if (i < 7)
34.            i++;
```

```
35.          else
36.             i = 0;
37.          tmp = i << 1;
38.          tmp &= 0b00001110;
39.          PORTB = (PORTB & 0b11110001) | tmp;
40.         }
41.      else if (!Button_is_pressd(BUTTON1))
42.               blockFlag = FALSE;
43.      delay_ms(5);
44.     }
45. }
46. //Συνάρτηση ελέγχου πατήματος κουμπιού
47. u08 Button_is_pressd(u08 ButtonId)
48. {
49.  u08 tmp;
50.  tmp = FALSE;
51.  if (!(PORTB & (1 << ButtonId)))
52.     {
53.      delay_ms(DEBOUNCE_TIME);
54.      if (!(PORTB & (1 << ButtonId)))
55.          tmp = TRUE;
56.     }
57.  return(tmp);
58. }
```

Υπάρχει video http://youtu.be/1JQghSLCiGs όπου παρουσιάζονται τα παραπάνω παραδείγματα.

Στο επόμενο παραγωγή ήχου με τον PIC και χρήση διακοπών (interrupts) και χρονιστή (timer).