ΕΜΠ - ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ & ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΑΘΗΝΑ 27 Νοεμβρίου 2015

Εργαστηριακές ασκήσεις στον Μικροελεγκτή AVR

3η Εργαστηριακή Άσκηση του AVR – Χρήση πληκτρολογίου

Στην άσκηση αυτή θα γίνει μελέτη της χρήσης του πληκτρολογίου 4×4.

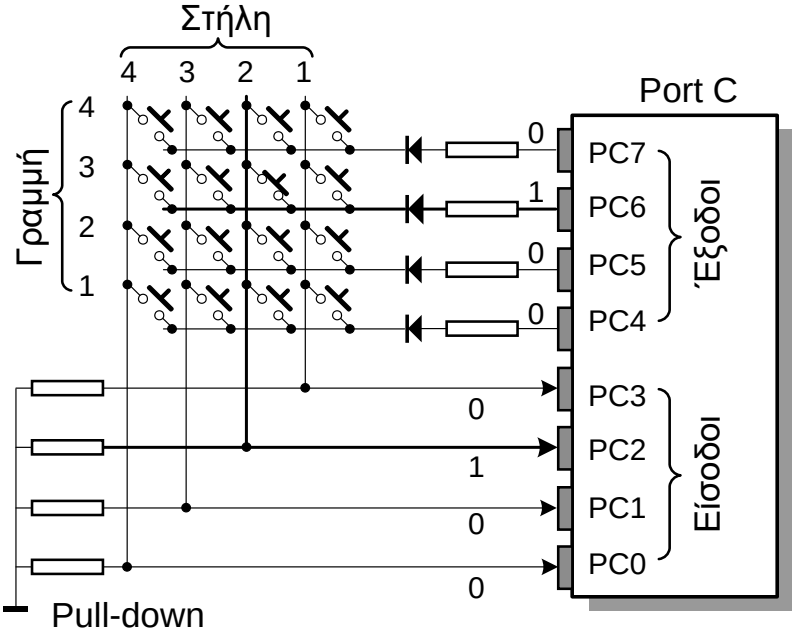
Έλεγχος Περιφερειακών Συσκευών

Η επικοινωνία ενός μικροελεγκτή με περιφερειακές συσκευές μπορεί να γίνει με διαφορετικούς τρόπους και πρωτόκολλα. Για κάθε διαφορετική περίπτωση και τεχνική επικοινωνίας, είναι ιδιαίτερα χρήσιμη η κατασκευή μιας βιβλιοθήκης λογισμικού (ρουτίνες) που αναλαμβάνουν όλες τις χαμηλού επιπέδου λειτουργίες (π.χ. τροφοδοσία και έλεγχος συγκεκριμένων ακροδεκτών, χρονισμός) που καθορίζονται από τον τρόπο επικοινωνίας, και παρέχουν υψηλότερου επιπέδου λειτουργίες (π.χ. αποστολή και λήψη χαρακτήρων), που χρησιμοποιούνται για την κατασκευή λογισμικού εφαρμογών. Οι βιβλιοθήκες αυτές ονομάζονται οδηγοί συσκευών. Για την κατασκευή τους είναι απαραίτητη η γνώση του τρόπου επικοινωνίας και της συνδεσμολογίας της περιφερειακής συσκευής. Συχνά, η περιφερειακή συσκευή συνοδεύεται από έναν ελεγκτή ο οποίος καθορίζει και υλοποιεί τον τρόπο επικοινωνίας. Τα τεχνικά του χαρακτηριστικά περιέχονται στο αντίστοιχο εγχειρίδιο τεχνικών προδιαγραφών. Η συνδεσμολογία προκύπτει από τα τεχνικά χαρακτηριστικά αλλά και τους περιορισμούς του συστήματος που θα συνδεθεί η περιφερειακή συσκευή (π.χ. διαθεσιμότητα ακροδεκτών).

Στην αναπτυξιακή πλακέτα EasyAVR6 διαθέτει μεταξύ των άλλων περιφερειακών συσκευών ένα πληκτρολόγιο 4×4 και μια οθόνη χαρακτήρων 2×16. Στην συνέχεια παρουσιάζονται οι συσκευές αυτές σε συνδυασμό με το αντίστοιχο λογισμικό οδήγησής τους.

Πληκτρολόγιο 4×4

Το πληκτρολόγιο 4×4 του αναπτυξιακού easyAVR6 αποτελείται από 16 πιεστικούς διακόπτες συνδεδεμένους όπως φαίνεται στο ακόλουθο σχήμα.



Η ανάγνωση του πληκτρολογίου γίνεται ανά γραμμή. Η γραμμή που επιθυμούμε να ελέγξουμε για πατημένους διακόπτες επιλέγεται θέτοντας έναν από τους ακροδέκτες PC7 – PC4 της θύρας PORTC του μικροελεγκτή στο λογικό 1. Οι 4 γραμμές του πληκτρολογίου επιλέγονται από τους ακροδέκτες PC7 – PC4 οι οποίοι πρέπει να είναι ρυθμισμένοι για έξοδο. Στη συνέχεια εντοπίζουμε τις στήλες των πιεσμένων διακοπτών, διαβάζοντας τον καταχωρητή PINC και ελέγχοντας τους ακροδέκτες PC3 – PC0 που τους έχουμε ρυθμίσει ως εισόδους. Λογικό 1 αντιστοιχεί σε πιεσμένο διακόπτη. Αν δεν υπάρχει πιεσμένος διακόπτης, λόγω των pull-down αντιστάσεων, έχουμε λογικό 0 στην είσοδο.

Η αρχικοποίηση της θύρα PORTC του μικροελεγκτή μπορεί να πραγματοποιηθεί με τις ακόλουθες εντολές (οι εσωτερικές αντιστάσεις pull-up πρέπει να είναι απενεργοποιημένες).

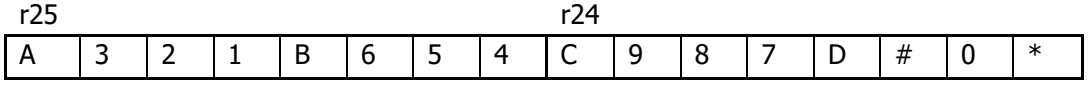
```
ldi r24 ,(1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)      ; θέτει ως εξόδους τα 4 MSB
out DDRC ,r24                                                  ; της θύρας PORTC
```

Για να ελέγξουμε μια γραμμή του πληκτρολογίου για πιεσμένους διακόπτες μπορούμε να χρησιμοποιήσουμε τη ρουτίνα scan_row. Στον καταχωρητή r24 αποθηκεύουμε τον αριθμό της γραμμής (1 – 4) που θέλουμε να ελέγξουμε και ανακτούμε το αποτέλεσμα του ελέγχου στα 4 λιγότερα σημαντικά bit του ίδιου καταχωρητή. Οι δύο εντολές *nop* μεταξύ της εντολής που επιλέγει τη γραμμή και της εντολής που διαβάζει τον καταχωρητή PINC είναι απαραίτητες καθώς μπορεί να απαιτηθούν μέχρι και δύο κύκλοι ρολογιού μέχρι μια αλλαγή στην κατάσταση των ακροδεκτών (λόγω της αδράνειας τους) να καταγραφεί στον καταχωρητή PINC.

Ρουτίνα: scan_row
Έλεγχος μιας γραμμής του πληκτρολογίου για πιεσμένους διακόπτες.
Είσοδος: Ο αριθμός της γραμμής προς ανάγνωση πρέπει να είναι αποθηκευμένος στον καταχωρητή r24 (τιμή 1-4).
Έξοδος: Στα 4 λιγότερο σημαντικά bit του r24 είναι αποθηκευμένη η κατάσταση κάθε διακόπτη.
Καταχωρητές: r25:r24
Καλούμενες υπορουτίνες: -

```
scan_row:
        ldi r25 ,0x08           ; αρχικοποίηση με '0000 1000'
back_:  lsl r25                 ; αριστερή ολίσθηση του '1' τόσες θέσεις
        dec r24                 ; όσος είναι ο αριθμός της γραμμής
        brne back_
        out PORTC ,r25          ; η αντίστοιχη γραμμή τίθεται στο λογικό '1'
        nop
        nop                     ; καθυστέρηση για να προλάβει να γίνει η αλλαγή κατάστασης
        in r24 ,PINC            ; επιστρέφουν οι θέσεις (στήλες) των διακοπτών που είναι πιεσμένοι
        andi r24 ,0x0f          ; απομονώνονται τα 4 LSB όπου τα '1' δείχνουν που είναι πατημένοι
        ret                     ; οι διακόπτες.
```

Τώρα που διαθέτουμε μια ρουτίνα που ελέγχει μια γραμμή του πληκτρολογίου για πιεσμένους διακόπτες μπορούμε να γράψουμε μια άλλη που θα την ενσωματώνει και θα ελέγχει ολόκληρο το πληκτρολόγιο. Το αποτέλεσμα του ελέγχου είναι ένας δυαδικός αριθμός με μήκος 16 bit που αποθηκεύεται στους καταχωρητές r25:r24. Κάθε διακόπτης από τους 16 αντιστοιχεί σε ένα bit των καταχωρητών r25:r24 όπου λογικό '1' σημαίνει ότι είναι πιεσμένος ο αντίστοιχος διακόπτης. Η αντιστοιχία φαίνεται στο επόμενο σχήμα.



Ρουτίνα: scan_keypad
Έλεγχος του πληκτρολογίου για πιεσμένους διακόπτες.
Είσοδος: καμία
Έξοδος: Στους καταχωρητές r25:r24 είναι αποθηκευμένη η κατάσταση των 16 διακοπτών του πληκτρολογίου.
Καταχωρητές: r27:r26, r25:r24
Καλούμενες υπορουτίνες: scan_row

```
scan_keypad:
        ldi r24 ,0x01           ; έλεγξε την πρώτη γραμμή του πληκτρολογίου
         rcall scan_row
        swap r24                ; αποθήκευσε το αποτέλεσμα
        mov r27 ,r24            ; στα 4 msb του r27
        ldi r24 ,0x02           ; έλεγξε τη δεύτερη γραμμή του πληκτρολογίου
        rcall scan_row
        add r27 ,r24            ; αποθήκευσε το αποτέλεσμα στα 4 lsb του r27
        ldi r24 ,0x03            ; έλεγξε την τρίτη γραμμή του πληκτρολογίου
        rcall scan_row
        swap r24                ; αποθήκευσε το αποτέλεσμα
        mov r26 ,r24            ; στα 4 msb του r26
        ldi r24 ,0x04           ; έλεγξε την τέταρτη γραμμή του πληκτρολογίου
        rcall scan_row
        add r26 ,r24            ; αποθήκευσε το αποτέλεσμα στα 4 lsb του r26
        movw r24 ,r26           ; μετέφερε το αποτέλεσμα στους καταχωρητές r25:r24
        ret
```

Έως τώρα έχουμε δει πώς να αρχικοποιήσουμε την θύρα PORTC του μικροελεγκτή για να χρησιμοποιήσουμε το πληκτρολόγιο και πώς να το ελέγξουμε για πιεσμένα πλήκτρα. Όμως αυτό που πραγματικά μας ενδιαφέρει δεν είναι ποιοί διακόπτες είναι πιεσμένοι τη στιγμή που ελέγχουμε το πληκτρολόγιο, αλλά ποιο πλήκτρο πάτησε ο χρήστης. Όταν ο χρήστης πατάει ένα πλήκτρο, αυτό μπορεί να μείνει πιεσμένο για αυθαίρετα μεγάλο χρονικό διάστημα. **Πρέπει να μπορούμε να ξεχωρίσουμε ποιο πλήκτρο πατήθηκε (ως ολοκληρωμένη ενέργεια) και όχι ποιο πλήκτρο είναι πατημένο.**

Για να το επιτύχουμε πρέπει να ελέγχουμε το πληκτρολόγιο για πιεσμένους διακόπτες με την ρουτίνα scan_keypad, **να αποθηκεύουμε το αποτέλεσμα στη μνήμη RAM του μικροελεγκτή** και στη συνέχεια με μια δεύτερη κλήση της ίδιας ρουτίνας να ξαναελέγξουμε το πληκτρολόγιο. Μια σύγκριση των δύο καταστάσεων του πληκτρολογίου θα αποκαλύψει τις διαφορές στην κατάσταση των διακοπτών. Το χρονικό διάστημα ανάμεσα στις διαδοχικές κλήσεις της συνάρτησης scan_keypad είναι κρίσιμο, διότι καθορίζει το χρόνο που θα πρέπει να μείνει πιεσμένος ένας διακόπτης από το χρήστη για να καταγραφεί από τον μικροελεγκτή. Αυτό σημαίνει ότι ένα μεγάλο χρονικό διάστημα μεταξύ των διαδοχικών κλήσεων της ρουτίνας θα αναγκάσει το χρήστη να κρατά πατημένο το πλήκτρο για αντίστοιχα μεγάλο χρονικό διάστημα (ώστε να μπορεί να αναγνωριστεί). Αντίθετα, ένα πολύ μικρό χρονικό διάστημα θα δημιουργήσει προβλήματα λόγω του σπινθηρισμού που παρουσιάζουν οι διακόπτες.

Η ρουτίνα scan_keypad_rising_edge υλοποιεί όσα αναφέρθηκαν πρωτύτερα και παράλληλα αντιμετωπίζει αποτελεσματικά το ζήτημα του σπινθηρισμού των διακοπτών.

Ρουτίνα: scan_keypad_rising_edge
Έλεγχος του πληκτρολογίου για διακόπτες που δεν ήταν πιεσμένοι την τελευταία φορά που κλήθηκε η ρουτίνα και τώρα είναι.
Είσοδος: Ο αναμενόμενος χρόνος σπινθηρισμού των διακοπτών σε ms είναι αποθηκευμένος στον καταχωρητή r24.
Έξοδος: Ένας αριθμός των 16 bit, ενδεικτικός των διακοπτών που «μόλις» πατήθηκαν, είναι αποθηκευμένος στους καταχωρητές r25:r24
Καταχωρητές: r27:r26, r25:r24, r23:r22
Καλούμενες υπορουτίνες: scan_keypad, wait_msec

Παρατηρήσεις:
Για να καταγραφούν οι διακόπτες που έχουν «μόλις» πιεστεί, η ρουτίνα χρησιμοποιεί την μεταβλητή _tmp_. Επειδή η μεταβλητή _tmp_ βρίσκεται στην RAM του μικροελεγκτή δεν είναι αρχικοποιημένη. Ο χρήστης πρέπει να την αρχικοποιήσει είτε ρητά είτε με μια κλήση της ρουτίνας μόνο για αυτό το σκοπό.

```
        ; ---- Αρχή τμήματος δεδομένων

.DSEG
_tmp_: .byte 2

        ; ---- Τέλος τμήματος δεδομένων
```

```
.CSEG

scan_keypad_rising_edge:
        mov r22 ,r24            ; αποθήκευσε το χρόνο σπινθηρισμού στον r22
        rcall scan_keypad       ; έλεγξε το πληκτρολόγιο για πιεσμένους διακόπτες
        push r24                ; και αποθήκευσε το αποτέλεσμα
        push r25
        mov r24 ,r22            ; καθυστέρησε r22 ms (τυπικές τιμές 10-20 msec που καθορίζεται από τον
        ldi r25 ,0              ; κατασκευαστή του πληκτρολογίου – χρονοδιάρκεια σπινθηρισμών)
        rcall wait_msec
        rcall scan_keypad       ; έλεγξε το πληκτρολόγιο ξανά και
        pop r23                 ; απόρριψε όσα πλήκτρα εμφανίζουν
        pop r22                 ; σπινθηρισμό
        and r24 ,r22
        and r25 ,r23
        ldi r26 ,low(_tmp_)     ; φόρτωσε την κατάσταση των διακοπτών στην
        ldi r27 ,high(_tmp_)    ; προηγούμενη κλήση της ρουτίνας στους r27:r26
        ld r23 ,X+
        ld r22 ,X
        st X ,r24               ; αποθήκευσε στη RAM τη νέα κατάσταση
        st -X ,r25              ; των διακοπτών
        com r23
        com r22                 ; βρες τους διακόπτες που έχουν «μόλις» πατηθεί
        and r24 ,r22
        and r25 ,r23
        ret
```

Τέλος, επειδή η κατάσταση του πληκτρολογίου στην μορφή ενός δυαδικού αριθμού δεν είναι ιδιαίτερα χρήσιμη υπάρχει και η ρουτίνα keypad_to_ascii που εντοπίζει τον διακόπτη που έχει πατηθεί και επιστρέφει τον κωδικό ascii του χαρακτήρα που αντιστοιχεί στον διακόπτη. Αν δεν είναι πιεσμένος κανένας διακόπτης επιστρέφει την τιμή 0, ενώ εάν είναι πατημένοι πολλοί επιστρέφει μόνο έναν από αυτούς (ο $1^{ος}$ που εντοπίζεται με βάση την σειρά εξερεύνησης των εντολών της ρουτίνας που ακολουθεί).

Ρουτίνα: keypad_to_ascii
Αντιστοίχιση διακοπτών, κωδικών ascii.
Είσοδος: Στους καταχωρητές r25:r24 είναι αποθηκευμένος ένας αριθμός 16 bit, ενδεικτικός της κατάστασης κάθε διακόπτη.
Έξοδος: Ο κωδικός ascii, που αντιστοιχεί στον πρώτο πατημένο διακόπτη που εντοπίστηκε, αποθηκεύεται στον καταχωρητή r24 ή 0 αν δεν έχει πατηθεί κάποιος.
Καταχωρητές: r27:r26, r25:r24
Παρατηρήσεις: Ο αριθμός των 16 bit που αποθηκεύεται στους καταχωρητές r25:r24 κατά την κλήση της ρουτίνας πρέπει να προέρχεται από μια εκ των scan_keypad ή scan_keypad_rising_edge.

```
keypad_to_ascii:        ; λογικό '1' στις θέσεις του καταχωρητή r26 δηλώνουν
        movw r26 ,r24   ; τα παρακάτω σύμβολα και αριθμούς
        ldi r24 ,'*'
        sbrc r26 ,0
        ret
        ldi r24 ,'0'
        sbrc r26 ,1
        ret
        ldi r24 ,'#'
        sbrc r26 ,2
        ret
        ldi r24 ,'D'
        sbrc r26 ,3     ; αν δεν είναι '1'παρακάμπτει την ret, αλλιώς (αν είναι '1')
        ret             ; επιστρέφει με τον καταχωρητή r24 την ASCII τιμή του D.
        ldi r24 ,'7'
        sbrc r26 ,4
        ret
```

r26

C	9	8	7	D	#	0	*

```
ldi r24 ,'8'
sbrc r26 ,5
ret
ldi r24 ,'9'
sbrc r26 ,6
ret
ldi r24 ,'C'
sbrc r26 ,7
 ret
ldi r24 ,'4'
sbrc r27 ,0
ret
ldi r24 ,'5'
sbrc r27 ,1
ret
ldi r24 ,'6'
sbrc r27 ,2
ret
ldi r24 ,'B'
sbrc r27 ,3
ret
ldi r24 ,'1'
sbrc r27 ,4
ret
ldi r24 ,'2'
sbrc r27 ,5
ret
ldi r24 ,'3'
sbrc r27 ,6
ret
ldi r24 ,'A'
sbrc r27 ,7
ret
clr r24
ret
```

; λογικό ‘1’ στις θέσεις του καταχωρητή r27 δηλώνουν
; τα παρακάτω σύμβολα και αριθμούς

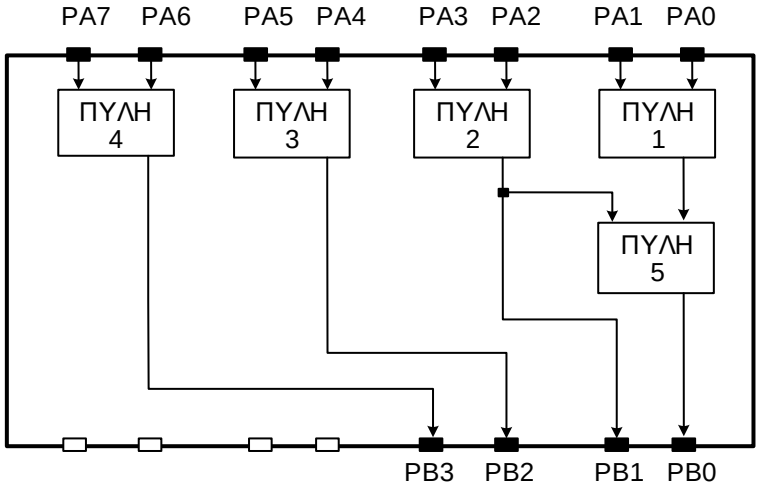
r27

A	3	2	1	B	6	5	4

Τα ζητούμενα της 3ης εργαστηριακής άσκησης του AVR

(Να γίνει επίδειξη στην αναπτυξιακή πλακέτα EasyAVR6 - αναμμένο led υποθέτουμε ότι αντιστοιχεί σε λογικό '1').

Ζήτημα 3.1 Να εξομοιωθεί η λειτουργία ενός υποθετικού I.C. με πύλες όπως φαίνεται στο σχήμα 3.1. Τα bits εισόδου πρέπει να αντιστοιχούν ακριβώς όπως φαίνονται στο σχήμα με τα dip switches PA7-0 και οι έξοδοι πρέπει να είναι τα LEDs PB3-0 (τα υπόλοιπα PB7-4 να είναι σβηστά). Οι πύλες 1, 2, 3, 4, 5 να υλοποιούν τις XOR, OR, NOR, NXOR και AND αντίστοιχα. Όσο πατάμε κάποιο από τα push buttons PC7-0 να αντιστρέφεται η ένδειξη του αντίστοιχου LED PB7-0. Αρχικά όλα τα led να είναι σβηστά και όλες οι αλλαγές να γίνονται κατά το πάτημα των push buttons. Το πρόγραμμα να δοθεί σε assembly.



Σχήμα 3.1 Υποθετικό I.C. με πύλες

Ζήτημα 3.2 Να υλοποιηθούν σε ένα σύστημα μικροελεγκτή AVR οι λογικές συναρτήσεις:

F0= (ABC + CD + DE)' , F1=AB+CDE', F2= F0 + F1

και να εμφανιστούν στα τρία MSB της θύρας εξόδου PortC (7-5). Οι μεταβλητές εισόδου (A, B, C, D, E) υποθέτουμε ότι αντιστοιχούν στα bit της θύρας εισόδου PortA (A = PA0, B = PA1 κ.ο.κ). Το πρόγραμμα να δοθεί σε C.

Ζήτημα 3.3 Γράψτε ένα πρόγραμμα το οποίο ανάβει τα leds PB0-7 για 3 sec όταν πατηθούν στη σειρά δύο συγκεκριμένα πλήκτρα στο keypad 4×4 (π.χ. ο διψήφιος αριθμός της ομάδας σας). Ανεξάρτητα από το χρονικό διάστημα για το οποίο θα μείνει πατημένο ένα πλήκτρο, το πρόγραμμά σας θα πρέπει να θεωρεί ότι πατήθηκε μόνο μια φορά. Το πρόγραμμα αυτό θα μπορούσε να αποτελέσει τη βάση μιας «ηλεκτρονικής κλειδαριάς». Ξεκινήστε σχεδιάζοντας απαραίτητα το διάγραμμα ροής και μετά γράψτε το πρόγραμμα.