# Εργαστηριακές Ασκήσεις Κεφ. 15

**Μπαντής Αστέριος (Αυτ. XXXXXXX)**

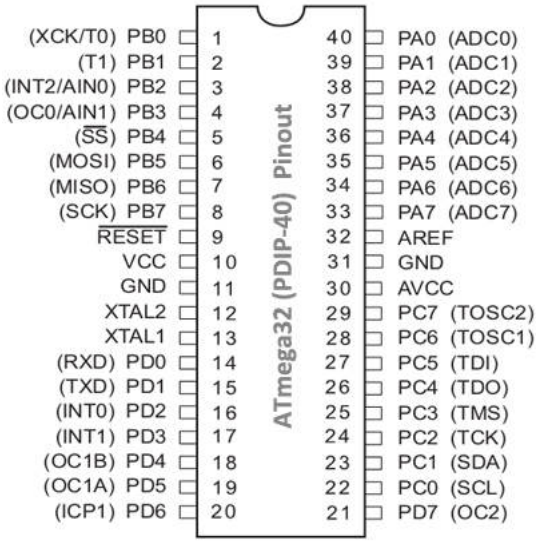
**2021-11-14**
**Σύνδεσμος:** censored

**Μικροελεγκτές**

## Περιεχόμενα

# 1 Εισαγωγή

Στο κεφάλαιο 15 του βιβλίου μαθαίνουμε πώς να χειριζόμαστε τις θύρες εισόδου και εξόδου (I/O) του μικροελεγκτή AVR ATMega32A. Είναι δυνατός ο χειρισμός ακροδεκτών είτε μεμονωμένα (δηλαδή όπως λειτουργεί η συνάρτηση pinmode(pinNumber, MODE) για το setup και η εντολή digitalWrite() και digitalRead() στο Arduino), είτε σε ομάδες των 8 ακροδεκτών, που ονομάζονται "θύρες" (ports). Ο μικροελεγκτής ATMega32A διαθέτει τέσσερις




Εικόνα 1: Το διαγραμμα pinout του μικροελεγκτή ATMega32. (Πηγή: atmega32-avr.com/atmega-32-pinout/)

τέτοιες θύρες: την PORTA, PORTB, PORTC, και PORTD. Ο έλεγχος ενός μεμονωμένου ακροδέκτη γίνεται με την εντολή `SBI ioregisterx, pinNumber` (Set Bit in I/O) και `CBI ioregisterx, pinNumber` (Clear Bit in I/O). Ο χειρισμός ολόκληρης της θύρας γίνεται με την εντολή `OUT ioregisterx, internalregister`. Οι καταχωρητές εισόδου και εξόδου που πρέπει να ελέγχουμε για κάθε θύρα στο setup είναι δύο: ο DDRx (Data Direction Register) και ο PORTx. Ο DDRx δηλώνει ποιοί ακροδέκτες σε μία θύρα είναι είσοδοι (με τον αριθμό "0") και ποιοί είναι έξοδοι (με τον αριθμό "1"). Ο PORTx για τους ακροδέκτες της εισόδου δηλώνει την ύπαρξη (1) ή απουσία (0) εσωτερικής pull-up αντίστασης, ενώ για την έξοδο δηλώνει το επίπεδο της τάσης του ακροδέκτη ("HIGH" και "LOW" στο Arduino). Υπάρχει επίσης ο ακροδέκτης PINx, από τον οποίο διαβάζουμε τα περιεχόμενα της εισόδου.

# 2 Άσκηση 15.1

Συνδέστε ένα dip switch 8 ακροδεκτών στη θύρα A και 8 LED στη θύρα B. Γράψτε ένα πρόγραμμα που να εμφανίζει στα LED την κατάσταση του dip switch.

## 2.1 Πρόγραμμα

```
.include "m32def.inc"

LDI R16,0xFF
OUT DDRB,R16 ;κάνε όλους τους ακροδέκτες της θύρας B εξόδους
LDI R16,0x00
OUT DDRA,R16 ;κάνε όλους τους ακροδέκτες της θύρας A εισόδους
LDI R16,0xFF
OUT PORTA,R16 ;πρόσθεσε εσωτ. αντιστάσεις pull-up σε όλες τις εισόδους

loop:
IN R16,PINA ;διάβασε την είσοδο και γράψε το αποτέλεσμα στον R16
OUT PORTB,R16 ;γράψε στην έξοδο την κατάσταση του R16
RJMP loop ;επανάλαβε
```

## 2.2 Επεξήγηση

Έχουμε συνδέσει τη μία μεριά του dip switch στους ακροδέκτες PA0-PA7 και την άλλη μεριά στη γείωση. Έχουμε συνδέσει τους ακροδέκτες PB0-PB7 σε σειρά με μια αντίσταση των 150Ω και ένα μπλε LED ανά pin. Προφανώς κάθε διακοπτάκι του dip switch είναι σε διαφορετικό ακροδέκτη της θύρας A και κάθε ζευγάρι αντίστασης-LED σε διαφορετικό ακροδέκτη της θύρας B. Τα LEDs καταλήγουν στη γείωση.
Στο setup ρυθμίζουμε τους ακροδέκτες της θύρας A σαν εισόδους με pull-up και τους ακροδέκτες της θύρας B σαν εξόδους. Στο περιβάλλον του Arduino αυτό θα μπορούσε να αντικατασταθεί με 16 εντολές pinMode() (π.χ.:`pinMode(1, INPUT_PULLUP);` ή `pinMode(2, INPUT_PULLUP);`...) ή με τρείς εντολές που χειρίζονται τους ίδιους καταχωρητές, π.χ. `DDRB=B11111111; DDRA=B00000000; PORTA=B11111111`.
Στο loop διαβάζουμε επανειλημμένα την κατάσταση της θύρας A μέσω του PINA και γράφουμε την κατάστασή της στον PORTB της θύρας B. Στο περιβάλλον του Arduino αυτό θα μπορούσε να αντικατασταθεί με 8 εντολές digitalRead() και 8 εντολές digitalWrite(), ή απλά εξισώνοντας τον καταχωρητή PORTB με το περιεχόμενο του καταχωρητή PINA.

# 3 Άσκηση 15.2

Συνδέστε 2 μπουτόν και 1 LED σε γραμμές της θύρας C. Γράψτε ένα πρόγραμμα που να ανάβει το LED όση ώρα και τα 2 μπουτόν είναι πιεσμένα.

## 3.1 Πρόγραμμα

```
.include "m32def.inc"
CBI DDRC, 7 ;pinMode(C7, INPUT_PULLUP);
SBI PORTC, 7

CBI DDRC, 6 ;pinMode(C6, INPUT_PULLUP);
SBI PORTC, 6

SBI DDRC, 0 ;pinMode(C0, OUTPUT);

loop:
SBIC PINC, 7 ;Check if button 1 is pressed
RJMP clearled

SBIC PINC, 6 ;check if button 2 is pressed
RJMP clearled

SBI PORTC, 0 ;if both are pressed, turn on the LED.
RJMP loop

clearled:
CBI PORTC, 0 ;If at least one is not pressed, clear the LED
RJMP loop
```

## 3.2 Επεξήγηση

Το κὺκλωμα είναι απλό. Στα pins C7 και C6 έχουμε συνδέσει τη μία άκρη του μπουτόν Προφανώς, ένα μπουτόν ανά ακροδέκτη. Η άλλη άκρη του μπουτόν είναι συνδεμένη στη γείωση. Στον ακροδέκτη C0 έχουμε συνδέσει ένα LED σε σειρά με μια αντίσταση. Αυτό καταλήγει επίσης στη γείωση. Ο λόγος που δε χρησιμοποιούμε κάποια άλλα pins είναι επειδή τα μεσαία pins της θύρας C (C2, C3, C4, C5) χρησιμοποιούνται ήδη από το JTAG όταν το fuse "JTAGEN" είναι ενεργοποιημένο. Το πρόγραμμα είναι απλό: Στην αρχή ορίζουμε τους ακροδέκτες της εισόδου (με pull-up) και τον ακροδέκτη της εξόδου. Έπειτα, ελέγχουμε περιοδικά αν τα μπουτόν είναι πατημένα. Εάν είναι, το LED ανάβει και γυρνάει ξανά στην αρχή του loop. Εάν έστω και ένα δεν είναι πατημένο, το πρόγραμμα μπαίνει στην υπορουτίνα clearled (ή vale0, όπως γράφτηκε στην τάξη), απενεργοποιεί το LED, και γυρνάει στην αρχή του loop.

# 4 Άσκηση 15.3

Γράψτε ένα πρόγραμμα ώστε στην αρχικοποίηση να αντιγράφονται τα dip switch στα LED. Στον κυρίως βρόχο, ο αριθμός που εμφανίζεται στα LED να αυξάνεται κατά 1 με το ένα μπουτόν και να μειώνεται [κατά 1] με το άλλο.

## 4.1 Πρόγραμμα

```
.include "m32def.inc"
;ίδιο setup με την άσκηση 15.1. Στο setup προσθέσαμε και το περιεχόμενο του loop
LDI R16,0xFF ;επειδή θέλουμε να τρέξει μονάχα μία φορά
OUT DDRB,R16
LDI R16,0x00 ;κάνε τη θύρα B έξοδο
OUT DDRA,R16
LDI R16,0xFF ;κάνε τη θύρα A είσοδο
OUT PORTA,R16 ;με pull-up
IN R16,PINA ;γράψε κατάσταση dip switch στον R16
OUT PORTB,R16 ;αντιγραφή dip switch στα LEDs

;ίδιο setup με την άσκηση 15.2
CBI DDRC, 7
SBI PORTC, 7
CBI DDRC, 6
SBI PORTC, 6

loop:
SBIS PINC, 7 ;αν το μπουτόν 1 είναι πατημένο, αύξησε τον αριθμό κατά ένα.
RJMP auksise ;αν όχι, έλεγξε το μπουτόν 2
SBIS PINC, 6 ;αν είναι πατημένο, μείωσε τον αριθμό κατά ένα
RJMP meiwse
RJMP loop ;αν όχι, γύρνα στην αρχή

auksise:
INC R16 ;αύξησε τον αριθμό στον R16 και πήγαινε στο eksodos:
eksodos:
OUT PORTB, R16 ;γράψε την κατάσταση στα LED
perimene:
SBIS PINC, 7 ;αν ακόμα είναι πατημένο τουλάχιστον ένα από τα δύο, περίμενε
RJMP perimene
SBIS PINC, 6
RJMP perimene
RJMP loop ;αν όχι, γύρνα στην αρχή.
```

```
meiwse:
DEC R16 ;μείωσε τον αριθμό στον R16 και γύρνα στο eksodos:
RJMP eksodos
```

### 4.2 Επεξήγηση

Το κύκλωμα είναι σχεδόν ίδιο με τον συνδυασμό του κυκλώματος της άσκησης 15.1 και της άσκησης 15.2, με μοναδική διαφορά πως πλέον δε χρησιμοποιούμε τον ακροδέκτη C0 για κάποιο LED. Στο πρόγραμμα έχουμε προσθέσει σχεδόν όλο το πρόγραμμα 15.1, έχοντας αφαιρέσει μονάχα το `loop:` και το `RJMP loop`. Επίσης έχουμε προσθέσει το setup του προγράμματος 15.2, έχοντας αφαιρέσει τη δήλωση του C0 ως είσοδο. Στο κυρίως πρόγραμμα ακολουθούμε παρόμοια μεθοδολογία με το πρόγραμμα 15.2, επειδή ελέγχουμε επανειλημμένα αν τα μπουτόν είναι πατημένα. Ανάλογα με το ποιό μπουτόν είναι πατημένο, καλούμε την αντίστοιχη υπορουτίνα (auksise ή meiwse). Για να μην αλλάζει συνέχεια κατάσταση το πρόγραμμα, αφού ένας κύκλος μηχανής έχει τη διάρκεια των 6.25μs, κάθε φορά που πατάμε ένα μπουτόν μπαίνουμε στην υπορουτίνα perimene μέχρι να μην πατιέται κανένα μπουτόν.
Μερικές φορές το περιεχόμενο του καταχωρητή R16, και κατά συνέπεια και των LEDs, ενδέχεται να αυξηθεί κατά 2 ή κατά 3. Αυτό συμβαίνει λόγω του debouncing των μηχανικών επαφών. Η αντιμετώπισή του δεν αποτελεί τμήμα της άσκησης, όμως το έχουμε αντιμετωπίσει στο μάθημα των ενσωματωμένων συστημάτων με πολλούς τρόπους (ολοκληρωμένα με latches, RC φίλτρα, χρονιστές, κ.α.).

## 5 Βιβλιογραφία

[1] Νικολαΐδης Νικόλαος, ΜΙΚΡΟΕΛΕΓΚΤΕΣ: Ασκήσεις, Πειράματα και Εφαρμογές με τον ATmega32, 2018