Εργαστηριακές Ασκήσεις Κεφ. 16

Μπαντής Αστέριος (Αυτ. XXXXXXX)

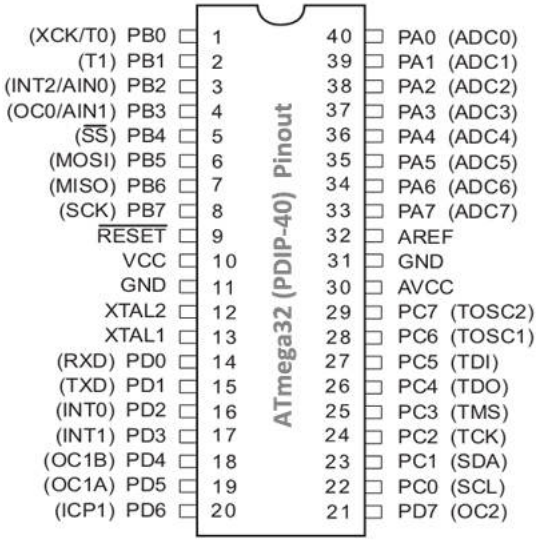
2021-11-21
Σύνδεσμος: censored

Μικροελεγκτές

Περιεχόμενα

1 Εισαγωγή

Στο κεφάλαιο 16 μαθαίνουμε τι είναι οι διακοπές (interrupts) και πώς χειριζόμαστε τις εξωτερικές διακοπές στον ATMega32A. Στο κεφάλαιο αυτό *δεν* ασχολούμαστε με τα pin change interrupts, τα οποία τα έχουμε ήδη εξετάσει στο μάθημα των Ενσωματωμένων Συστημάτων. Οι διακοπές είναι ένας τρόπος "χειρισμού" του προγράμματος από κάποιο εξωτερικό περιφερειακό. Μέσω των διακοπών αυτών δίνεται η δυνατότητα σε κάποιο περιφερειακό να είναι υπεύθυνο για την εκτέλεση κάποιας υπορουτίνας στον μικροελεγκτή, η οποία ρουτίνα δεν εκτελείται στο κύριο μέρος του προγράμματος. Μπορούμε, δηλαδή, πατώντας ένα πλήκτρο ή δίνοντας σήμα με άλλον τρόπο σε έναν ακροδέκτη του μικροελεγκτή, να διακόψουμε το κυρίως πρόγραμμα σε όποιο σημείο βρίσκεται[1], να σώσουμε στον stack pointer το περιεχόμενο του SREG και τη θέση του προγράμματος, να εκτελέσουμε την υπορουτίνα που είναι προσβάσιμη μόνο μέσω του interrupt, και να γυρίσουμε στο κυρίως πρόγραμμα χωρίς να έχουμε αλλοιώσει κανένα από τα δεδομένα μας. Τα interrupts είναι ιδιαίτερα χρήσιμα για χρονικά κρίσιμες υπηρεσίες, δηλαδή είτε βραχύχρονα σήματα που ενδέχεται να σταλούν στον μΕ την ώρα που εκείνος δε διαβάζει τους αντίστοιχους ακροδέκτες στο κυρίως πρόγραμμα με την εντολή `IN`, είτε σήματα χρονισμού, είτε κάποια υπηρεσία έκτακτης ανάγκης, όπως ένα πλήκτρο emergency stop. Ο μικροελεγκ-



Εικόνα 1: Το διάγραμμα pinout του ATMega32A

τής ATMega32A διαθέτει 3 ακροδέκτες εξωτερικών διακοπών. Στις εξωτερικές διακοπές, ο μικροελεγκτής γνωρίζει ακριβώς ποιός ακροδέκτης/ποιό περιφεριακό προκάλεσε τη διακοπή. Οι ακροδέκτες αυτοί, για τον ATMega32A, είναι ο INT0, INT1, και INT2. Οι INT0 και INT1 μπορούν να ενεργοποιήσουν διακοπές είτε σε κατερχόμενη παρυφή (falling edge), είτε σε ανερχόμενη παρυφή (rising edge), είτε σε οποιαδήποτε αλλαγή παρυφής (falling & rising edge),

[1] Όταν τελειώσει την εκτέλεση της εντολής που έτρεχε τη στιγμή που δόθηκε το σήμα

είτε σε χαμηλή στάθμη (low). Ο INT2 υποστηρίζει μόνο τις δύο πρώτες περιπτώσεις, δηλαδή τις αλλαγές παρυφής, αλλά όχι τη χαμηλή στάθμη. Σε ακριβότερους μικροελεγκτές δίνεται επίσης η δυνατότητα interrupt σε υψηλή στάθμη. Στο Arduino μπορούμε να ορίσουμε external interrupts με τον εξής τρόπο:
`attachInterrupt(digitalPinToInterrupt(αριθμοςPin), υποροutίνα, επιλογή);` όπου αριθμόςPin=2, 3, ... και επιλογή = FALLING ή RISING ή CHANGE ή LOW.

2 Άσκηση 16.1

Συνδέστε ένα μπουτόν στην εξωτερική είσοδο διακοπής INT0 (ακροδέκτης PD2) και ενα LED σε μία γραμμή της θύρας C. Γράψτε τις απαραίτητες αρχικοποιησεις για ένα πρόγραμμα που θα αλλάζει κατάσταση στο LED μόλις πιέζεται το μπουτόν. Οι εργασίες που πρέπει να γίνουν σχετικά με τη διακοπή είναι, ρύθμιση του διανύσματος διακοπής, ρύθμιση της διακοπής σε κατερχόμενη παρυφή, ενεργοποιηση της διακοπής INT0, ενεργοποίηση των διακοπών της CPU, συγγραφή της ρουτίνας διακοπής η οποία ανατρέπει σε κάθε της κλήση την κατάσταση του LED.

2.1 Πρόγραμμα

```
.include "m32def.inc"

    .org 0
    RJMP reset ;πέρνα πάνω από το διάνυσμα interrupt του INT0

    .org INT0addr ;αρχικοποίηση interrupt INT0
    RJMP INT0_irqRout ;πάνε στην υποροutίνα αν έρθει σήμα interrupt

reset:
    LDI R16,low(RAMEND)
    OUT SPL,R16 ;αρχικοποίηση του stack pointer
    LDI R16,high(RAMEND)
    OUT SPH,R16

    CBI DDRD,2 ;αρχικοποίηση INT0 ως είσοδος με pull-up
    SBI PORTD,2

    SBI DDRC,0 ;αρχικοποίηση C0 ως έξοδος

    LDI R16,(1<<ISC01)|(0<<ISC00) ;0|0=low, 1|1=rising, 1|0=falling edge
    OUT MCUCR,R16 ;falling edge επιλογή -> MCUCR

    LDI R16,(1<<INTF0)
```

```
    OUT GIFR,R16 ;αλλαγή σημαίας διακοπής INT0

    LDI R16,(1<<INT0)
    OUT GICR,R16 ;ενεργοποίηση INT0 μέσω μάσκας

    SEI ;ενεργοποίηση interrupts (I flag, SREG)
main:
    NOP ;οποιοδήποτε πρόγραμμα εδώ
    RJMP main

INT0_irqRout:
    SBIC PINC,0 ;πρόγραμμα ενεργοποίησης/απενεργοποίησης LED
    CBI PORTC,0
    SBIS PINC,0
    SBI PORTC,0
    RETI ;Περίπου η λειτουργία RET *και* SEI στην ίδια εντολή
```

2.2 Επεξήγηση

Το κύκλωμα είναι απλό και ίδιο και για τις τρείς ασκήσεις. Στον ακροδέκτη INT0 συνδέουμε τον έναν ακροδέκτη ενός μπουτόν, και τον άλλο ακροδέκτη τον συνδέουμε στη γείωση. Για την έξοδο έχουμε μια αντίσταση και ένα LED, τα οποία καταλήγουν στη γείωση. Στο πρόγραμμα φροντίζουμε το διάνυσμα του INT0 (0x02 στο πρόγραμμα) να μη χρησιμοποιείται από το πρόγραμμα, άρα, ξεκινώντας από το 0x00 (Reset) πηγαίνουμε στη διεύθυνση 0x04. Αρχικοποιούμε τον stack pointer, επειδή χρησιμοποιείται για την επιστροφή στη σωστή διεύθυνση προγράμματος μετά την εκτέλεση της υπορουτίνας interrupt. Αρχικοποιούμε τις ακροδέκτες της εισόδου και της εξόδου και ξεκινάμε τη ρύθμιση του interrupt. Η αντίστοιχη εντολή στον Arduino θα ήταν η εξής:
`attachInterrupt(digitalPinToInterrupt(16), INT0_irqRout, FALLING);` Ρυθμίζουμε, αρχικά, τα bits ISC01 και ISC00 του MCUCR ώστε να ενεργοποιείται το interrupt σε κατερχόμενη παρυφή. Έπειτα αλλάζουμε τη σημαία διακοπής στο bit INTF0 του General Interrupt Flag Register και ενεργοποιούμε το INT0 στον General Interrupt Control Register μέσω μάσκας. Στη συνέχεια, ενεργοποιούμε το "I" (interrupt) flag του Status Register, για να μπορεί να δέχεται interrupt requests ο μικροελεγκτής από αυτό το σημείο του προγράμματος και μετά. Το αφήσαμε για το τέλος του setup σε περίπτωση που έτρεχε κάποιο interrupt το οποίο διακόψει το πρόγραμμα πριν ολοκληρώσουμε την αρχικοποίηση. Το κύριο πρόγραμμα μπορεί να περιέχει οτιδήποτε, αλλά δεν καλούμε την υπορουτίνα INT0_irqRout εκεί. Στην INT0_irqRout γράφουμε ένα απλό πρόγραμμα που ανάβει το LED αν είναι σβησμένο, ή σβήνει το LED αν εκείνο είναι ενεργό. Εν τέλει, γυρνάμε στο κυρίως πρόγραμμα με την εντολή RETI.

Παρατηρούμε πως το LED αλλάζει κατάσταση όταν πατάμε το μπουτόν, όχι όταν το αφή-

νουμε. Αυτό συμβαίνει επειδή ακολουθούμε τη συνδεσμολογία γείωση - μπουτόν - pullup/είσοδος. Όταν το μπουτόν δεν είναι πατημένο, η είσοδος βλέπει υψηλό λογικό επίπεδο (HIGH) λόγω της pull-up. Όταν πατάμε το μπουτόν, βραχυκυκλώνουμε με τη γείωση και βλέπουμε χαμηλή στάθμη (LOW).

3 Άσκηση 16.2

Συνεχίζοντας την άσκηση 16.1 κάντε τα εξης. α) Βάλτε σημείο παύσης (breakpoint - F9) στην πρώτη εντολή της ρουτινας διακοπής. Επιβεβαιώστε ότι κάθε φορά που πατατε το μπουτόν εκτελείται η ρουτίνα διακοπής. β) Εκτελώντας το πρόγραμμα βηματικά (F11) μετρήστε την ελάχιστη και τη μεγίστη χρονική καθυστέρηση σε κύκλους από την ώρα που πατάτε το μπουτόν μέχρις ότου έρθει η σειρά να εκτελεστεί η πρώτη εντολή της ρουτίνας διακοπής. Πού οφείλεται η διαφορά αυτή μεταξύ ελάχιστης και μέγιστης τιμής; Φροντίστε όταν πατάτε το μπουτόν να μην εκτελείται η ρουτίνα διακοπής. γ) Αντικαταστήστε την εντολή RETI με RET και εξηγήστε τη συμπεριφορά της CPU.

3.1 Πρόγραμμα

α) Ακριβώς το ίδιο πρόγραμμα με την άσκηση 16.1
β) Ακριβώς το ίδιο πρόγραμμα με την άσκηση 16.2
γ) Μοναδική αλλαγή η αντικατάσταση του RETI με RET. Για να επαναφέρουμε τα interrupts μπορούμε να γράψουμε SEI και μετά RET.

3.2 Επεξήγηση

Η άσκηση 16.2α επιβεβαιώθηκε στο βίντεο. Η ρουτίνα ενεργοποιείται κάθε φορά που πατάμε το μπουτόν.
Στην 16.2β παρατηρήσαμε πως ο ελάχιστος χρόνος είναι 7 κύκλοι μηχανής και ο μέγιστος χρόνος 8 κύκλοι μηχανής. Η διαφορά αυτή οφείλεται στον τρόπο που χειρίζεται τα interrupts ο μικροελεγκτής, δηλαδή περιμένει να τελειώσει η προηγούμενη εντολή για να μεταφέρει το πρόγραμμα στη ρουτίνα interrupt. Η NOP εκτελείται σε έναν κύκλο μηχανής, ενώ η RJMP σε δύο κύκλους μηχανής. Έτσι, ανάλογα με το ποιά από τις δύο εντολές του κυρίου προγράμματος εκτελείται εκείνη τη στιγμή και ποιά είναι η επόμενη, έχουμε διαφορά ενός κύκλου μηχανής.
Στην άσκηση 16.3 παρατηρούμε πως το πρόγραμμα εκτελεί την INT0_irqRout μονάχα μία φορά. Δηλαδή, αναγνωρίζει μονάχα το πρώτο πάτημα του μπουτόν ως αίτημα διακοπής. Εάν παρατηρήσουμε τον status register, ο λόγος είναι προφανής: μπαίνοντας στην υπορουτίνα INT0_irqRout ο μικροελεγκτής απενεργοποιεί το Interrupt flag του SREG, το οποίο ενεργοποιείται στο τέλος της εντολής με την εντολή RETI. Έτσι, πλέον ο μικροελεγκτής μας δεν είναι σε θέση να λάβει νέα interrupt requests, επειδή το I flag είναι ανενεργό! Αντικαθιστώντας το RETI με SEI και RET μπορούμε να προσομοιώσουμε τη λειτουργεία του RETI σε κάποιο βαθμό.

Παρατηρήσαμε όμως, στο εργαστηριακό μάθημα της Τρίτης, πως όταν εφαρμόστηκε αυτή η λύση, ο μικροελεγκτής έμπαινε στη ρουτίνα interrupt δύο φορές ανά πάτημα μπουτόν.

4 Άσκηση 16.3

Συνεχίζοντας την άσκηση 16.2 αλλάξτε τη διακοπή από κατερχόμενη παρυφή σε χαμηλή στάθμη. Καταγράψτε και ερμηνεύστε τις παρατηρήσεις σας. Επαναλάβετε με ενεργή παρυφή την ανερχόμενη.

4.1 Πρόγραμμα

Οι μόνες αλλαγές γίνονται στις 2 γραμμές που σχετίζονται με τον MCUCR. Συγκεκριμένα:

```
;Για falling edge
  LDI R16,(1<<ISC01)|(0<<ISC00)
  OUT MCUCR,R16

;Για rising edge
  LDI R16,(1<<ISC01)|(1<<ISC00)
  OUT MCUCR,R16

;Για χαμηλή στάθμη
  LDI R16,(0<<ISC01)|(0<<ISC00) ;λεπτομέρειες: πινακάκι στο βιβλίο
  OUT MCUCR,R16
```

4.2 Επεξήγηση

Στην περίπτωση της ανερχόμενης παρυφής, το πρόγραμμα λειτουργεί κανονικά και το LED αλλάζει κατάσταση όταν αφήνουμε το μπουτόν. Στην περίπτωση της χαμηλής στάθμης, όταν το πλήκτρο είναι πατημένο, η φωτεινότητα του LED φαίνεται να πέφτει. Αυτό που πραγματικά συμβαίνει είναι πως αναβοσβήνει πολύ γρήγορα, αλλάζοντας κατάσταση συνεχώς για όσο κρατάμε το μπουτόν πατημένο. Το ανθρώπινο μάτι αντιλαμβάνεται αυτή τη συχνότητα ενεργοποίησης-απενεργοποίησης του LED ως χαμηλότερη ένταση φωτεινότητας. Το φαινόμενο αυτό μπορεί να αντιμετωπιστεί με πολύ λίγες αλλαγές, π.χ. εάν χρησιμοποιήσουμε κάποιον καταχωρητή σαν μεταβλητή "isButtonStillPressed" που να μας βγάζει έξω από τη ρουτίνα interrupt με RETI εάν το μπουτόν είναι ακόμη πατημένο, ή τροποποιώντας τη ρουτίνα `perimene` από τις ασκήσεις του κεφαλαίου 15 ώστε να λειτουργεί για ένα μπουτόν, μένοντας μέσα στη ρουτίνα interrupt για όσο χρονικό διάστημα η στάθμη του INT0 είναι χαμηλή.

5 Erratum

Στην προηγούμενη αναφορά έγραψα (λανθασμένα) πως ο κύκλος μηχανής στα 16MHz είναι 6.25μs. Το σωστό είναι 62.5ns, επειδή $\frac{1}{16\texttt{MHz}} = 62.5\texttt{ns}$. Ο χρόνος των 6.25μs θα ίσχυε για 160kHz, ενώ για τα 8MHz του εσωτερικού κρυστάλλου του ATMega32A είναι 125ns

6 Βιβλιογραφία

[1] Νικολαΐδης Νικόλαος, ΜΙΚΡΟΕΛΕΓΚΤΕΣ: Ασκήσεις, Πειράματα και Εφαρμογές με τον ATmega32, 2018
[2] Νικολαΐδης Νικόλαος, ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ, 2020