Εργαστηριακές Ασκήσεις Κεφ. 17

Μπαντής Αστέριος (Αυτ. XXXXXXX)

2021-11-28
Σύνδεσμος: censored

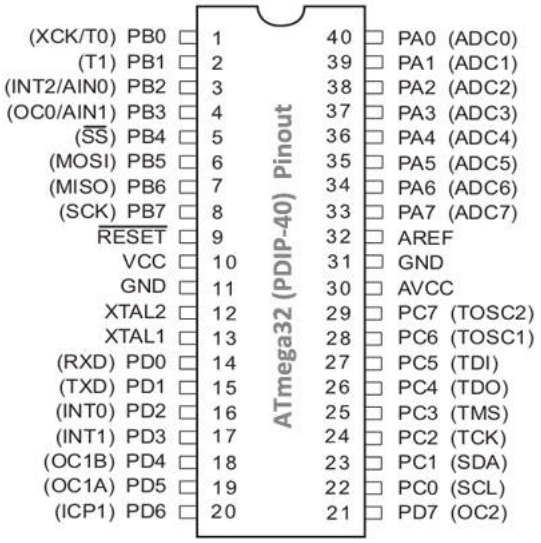
Μικροελεγκτές

Περιεχόμενα

1 Εισαγωγή

Στο κεφάλαιο 17 μαθαίνουμε τον τρόπο χειρισμού των χρονιστών των 8 bits (T0, T2) του ATMega32. Οι χρονιστές αυτοί είναι χρήσιμοι για την εκτέλεση περιοδικών διεργασιών, τη δημιουργία κυματομορφών, τη μέτρηση παλμών, τη δημιουργία PWM σήματος, την αποφυγή εισαγωγής υπορουτίνας τύπου delay στο πρόγραμμα, και άλλα. Οι πιο χρήσιμοι καταχωρητές που αφορούν τους χρονιστές είναι ο TCCRx (Timer/Counter Control Register), ο OCRx (Output Compare Register), ο TCNTx (Timer Counter Register), ο TIFRx (Timer Interrupt Flag Register), και ο TIMSK (Timer Interrupt Mask Register), ενώ υπάρχουν και άλλοι, όπως ο SFIOR (Special Function Input Output Register).

Για τον χρονιστή T0, ο καταχωρητής TCCR0 περιέχει ρυθμίσεις όπως υποδιαίρεση ρολογιού (prescaler) και χρήση εξωτερικού ρολογιού (CS02, CS01, CS00 bits), επιλογή του mode λειτουργείας, π.χ. κανονική λειτουργία, CTC, fast PWM, phase-correct PWM (WGM01, WGM00 bits), και χειρισμό του ακροδέκτη OC0 (PB3) στη δημιουργία timer interrupt (COM01, COM00 bits). Ο OCR0 περιέχει την τιμή που πρέπει να φτάσει ο χρονιστής για να δώσει timer interrupt. Ο TIFR περιέχει τα interrupt flags, τα οποία αλλάζουν κατάσταση όταν ο μετρητής φτάσει στον αριθμό που πρέπει να μετρήσει, δηλ. το περιεχόμενο του OCR0.



Εικόνα 1: Το διάγραμμα pinout του ATMega32A

2 Άσκηση 17.1

α) Γράψτε το πρόγραμμα της σελίδας του βιβλίου. Τρέξτε το βηματικά με το πλήκτρο F11 και παρατηρήστε τους καταχωρητές που σχετίζονται με τον χρονιστή T0.

β) Κάντε 1 το bit WGM01 και συνεχίστε τη βηματική εκτέλεση του προγράμματος και την παρατήρηση των καταχωρητών του T0.
γ) Κάντε 1 και το bit COM00 και ανοίξτε ώστε να παρακολουθείτε και τη θύρα B. Κάντε εξόδους τις γραμμές της και επαναλάβετε το βήμα (β).

2.1 Πρόγραμμα

```
.include "m32def.inc"
.def work=R16

    SER work
    OUT DDRB,work ;ενεργοποίηση θύρας B
    LDI work,16
    OUT OCR0,work ;μέτρα μέχρι το 16
    LDI work,(1<<CS00)|(1<<WGM01)|(1<<COM00) ;α|β|γ
    OUT TCCR0,work ;prescaler=1, CTC mode, αντιστρ. B3

loop:
    NOP
    RJMP loop
```

2.2 Επεξήγηση

Παρατηρούμε πως στο πρόγραμμα του βιβλίου δεν ενεργοποιείται ποτέ το interrupt στον αριθμό 16 (0x10), επειδή ο TIFR αλλάζει κατάσταση πριν δοθεί τιμή στον OCR0, δηλαδή όσο εκείνος είναι μηδέν, ενώ μετά δεν καθαρίζεται ποτέ το περιεχόμενό του. Βάζοντας την τιμή στον καταχωρητή OCR0 στον simulator πριν αλλάξουμε τις ρυθμίσεις στον OCR0, το πρόβλημα λύνεται χωρίς τον καθαρισμό του TIFR. Βέβαια, ο καθαρισμός των καταχωρητών πριν τη χρήση τους είναι μια καλή πρακτική, επειδή μπορεί να έχουν ήδη κάποιο περιεχόμενο από την προηγούμενη φορά που χρησιμοποιήθηκαν. Κατά τη διάρκεια της άσκησης, τα περιεχόμενα των TCCR0 και OCR0 μένουν σταθερά.

Ο TCCR0 περιέχει τα bits που εμείς ενεργοποιήσαμε (CS00 ώστε Prescaler=1, WGM01 ώστε να ενεργοποιηθεί το CTC mode, COM00 ώστε να αντιστρέφεται η κατάσταση του ακροδέκτη OC0 (PB3) σε κάθε μέτρηση. Ο OCR0 περιέχει τον αριθμό 16. Ο TCNT0 αυξάνεται κατά 1 σε κάθε κύκλο (όταν το βλέπουμε να αυξάνεται κατά δύο, στην πραγματικότητα έχουν περάσει 2 κύκλοι, επειδή δε μπορεί να διακόψει το πρόγραμμα ανάμεσα σε κάποια εντολή πολλών κύκλων, όπως η RJMP. Το bit OCF0 του TIFR ενεργοποιείται όταν ο μετρητής φτάσει στο 16.

Στο κομμάτι (α) της άσκησης, ο χρονιστής μετράει από το 0x00 μέχρι το 0xFF κάθε φορά, παρόλο που η τιμή που μας ενδιαφέρει είναι η 0x10. Στο κομμάτι (β) της άσκησης ενεργοποιούμε το bit WGM01 και πλέον ο TCNT μετράει μονάχα από το 0x00 μέχρι το 0x10. Τέλος, στο

κομμάτι (γ) της άσκησης ενεργοποιούμε το bit COM00. Έτσι, κάθε φορά που ο μετρητής φτάνει τον αριθμό 0x10, η κατάσταση του ακροδέκτη PB3 θα αντιστρέφει την κατάστασή του, δηλαδή από 0 θα γίνεται 1, και από 1 θα γίνεται 0.

3 Άσκηση 17.2

Τρέξτε μερικές φορές το πρόγραμμα του παραδείγματος 17.2 στον προσομοιωτή και μετρήστε την περίοδο της περιοδικής εργασίας. Αυξήστε κατά 1 την τιμή που φορτώνεται στον T0value προσομοιωτή και μετρήστε ξανά. Επαναλάβετε 12 φορές και καταγράψτε τις τιμές που βάζετε στον T0value και την περιοδο που μετράτε. Εξηγήστε τη συμπεριφορά του προγράμματος.

3.1 Πρόγραμμα

```
.include "m32def.inc"
.def work=R17
.def T0value=R18
.def stimer=R16

    LDI stimer,21 ;software timer
    LDI T0value,-188+5 ;επιθυμητή καθυστέρηση
    OUT TCNT0,T0value ;αρχ. τιμή T0, πρώτο τρέξιμο
    LDI work,(1<<TOV0)
    OUT TIFR,work ;μηδενισμός σήματος
    LDI work,(1<<CS00)
    OUT TCCR0,work ;Prescaler=1

loop:
    IN work,TIFR ;polling
    SBRS work,TOV0
    RJMP loop
    OUT TCNT0,T0value ;αρχ. τιμή T0 για όλες τις υπόλοιπες φορές
    LDI work,(1<<TOV0)
    OUT TIFR,work ;μηδενισμός σημαίας διακοπής
    DEC stimer
    BRNE loop
    LDI stimer,21
    NOP ;breakpoint
    NOP ;κάποια περιοδική εργασία ...
    NOP
    RJMP loop
```

3.2 Επεξήγηση

Τοποθετούμε το breakpoint στην πρώτη "NOP" εντολή του προγράμματος και τρέχουμε το πρόγραμμα n=1+12=13 φορές. Σε κάθε εκτέλεση του προγράμματος αυξάνουμε κατά ένα το περιεχόμενο του T0value, δηλαδή έχουμε `T0value` $= -188+5 = i$, όπου $i \in [0, 12] \in \mathbb{N}^*$. *Ξεκινάμε τη μέτρηση μετά τη λήξη του setup*, άρα αδειάζουμε το περιεχόμενο του Cycle Counter όταν φτάνουμε στο σημείο εκείνο. Τα αποτελέσματά μας είναι:

T0value	**Κύκλοι**
-188+5	3973
-188+6	3973
-188+7	3893
-188+8	3889
-188+9	3889
-188+10	3889
-188+11	3809
-188+12	3805
-188+13	3805
-188+14	3805
-188+15	3725
-188+16	3721
-188+17	3721

Παρατηρούμε πως κάποια αποτελέσματα μας δίνουν διαφορετικό αριθμό κύκλων, ενώ άλλα αποτελέσματα μας δίνουν τον ίδιο ακριβώς αριθμό κύκλων. Αυτό συμβαίνει επειδή, σε κάθε περίπτωση timer interrupt κατά τη διάρκεια εκτέλεσης του προγράμματος, το πρόγραμμα θα συνεχίσει την εκτέλεσή του μέχρι να ξαναβρεθεί στην εντολή IN στην αρχή του loop, ώστε το interrupt να διαβαστεί. Η παραπάνω ιδιαιτερότητα του προγράμματος, σε συνδυασμό με την ιδιότητα κάποιων εντολών να εκτελούνται σε πάνω από έναν κύκλο (RJMP:2 κύκλοι, SBRS:2/3 κύκλοι...) δίνουν ως αποτέλεσμα αυτές τις διαφοροποιήσεις και αυτές τις ομοιότητες σε κάθε περίπτωση.

4 Άσκηση 17.3 (Τροποποιημένη λόγω COVID-19)

Γράψτε ένα πρόγραμμα που να παράγει στη γραμμή PB.3 (OC0) τετραγωνική κυματομορφή συχνότητας 2 Hz. Επαληθεύστε με τον προσομοιωτή και ένα LED το οποίο θα συνδέσετε στη γραμμή PB.3.

4.1 Πρόγραμμα

```
.include "m32def.inc"

SBI DDRB,3
LDI R16,243
OUT OCR0,R16 ;επιθυμητή τιμή μέτρησης
LDI R16,(1<<CS00)|(1<<CS02)|(1<<COM00)|(1<<WGM01)
    ;P=1024, Fcpu=1MHz, αντιστρ. κατάστασης B3, CTC mode
OUT TCCR0,R16

loop:
RJMP loop
```

4.2 Επεξήγηση

Στην άσκηση αυτή, λόγω της έλλειψης παλμογράφου και beeper, πρέπει να προγραμματίσουμε τον μικροελεγκτή ώστε να χρησιμοποιεί τον Timer 0 για τη δημιουργία ενός τετραγωνικού κύματος συχνότητας 2Hz. Συνεπώς, το LED θα πρέπει να αναβοσβήνει 2 φορές το δευτερόλεπτο, άρα 60 φορές στα 30 δευτερόλεπτα. Θεωρώντας ως σημείο έναρξης τα 0ms, το LED θα πρέπει να ανάβει περιοδικά στα n+0 δευτερόλεπτα και n+0.5 δευτερόλεπτα, ενώ να σβήνει στα n+0.25 και n+0.75 δευτερόλεπτα αντίστοιχα. Άρα η αλλαγή κατάστασης γίνεται κάθε 250ms (4Hz). Για να φτάσουμε σε τόσο χαμηλές συχνότητες, αποσυνδέουμε το εξωτερικό ρολόι των 16MHz του μικροελεγκτή και ρυθμίζουμε το εσωτερικό στο 1MHz (6CK+4ms). Προφανώς είναι $n \in \mathbb{N}^*$. Υπολογίζουμε τα εξής μεγέθη:

$$f_{change} = 2f_{wave} = 4\texttt{Hz}$$

$$T_{change} = \frac{1}{f_{change}} = 0.25\texttt{s}$$

$$n_{cycles} = T_{change} \cdot f_{cpu} = 250000 \ \texttt{cycles}$$

Λόγω του μεγάλου αριθμού κύκλων και της χαμηλής συχνότητας, ξεκινάμε από τον μεγαλύτερο prescaler:

$$n_{cycles,scaled} = \frac{n_{cycles}}{P_{prescaler}} = \frac{250000}{1024} = 244.14 \ \texttt{cycles}$$

$$\texttt{OCR0} = \lfloor n_{cycles,scaled} \rfloor - 1 = 243$$

5 Βιβλιογραφία

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