Ενσωματωμένα Συστήματα Μικροεπεξεργαστών

Project 1

Ημερομηνία Παράδοσης: 16 Οκτοβρίου 2021

Αμπλιανίτης Κωνσταντίνος

2017030014

Σκοπός της Άσκησης:

Σκοπός της πρώτης εργαστηριακής άσκησης είναι να γίνει η πρώτη επαφή με την οικογένεια των μικροελεγκτών της AVR (πιο συγκεκριμμένα την οικογένεια της atmega 16) και το περιβάλλον με το οποίο θα διεξαχθούν τα υπόλοιπα εργαστήρια. Επιπλέον, μέσω της άσκησης γίνεται εισαγωγή στους timers/counters και στον τρόπο χρήσης τους. Ιδιαίτερη σημασία δίνεται στην χρήση του prescaler.

Περιγραφή της τεχνολογίας που χρησιμοποιήθηκε:

Για την υλοποίηση της συγκεκριμμένης εργαστηριακής άσκησης χρησιμοποιήθηκε το Microchip Sudio (version 7.0.2542) της Microchip Technology.

Περιγραφή επίλυσης της άσκησης:

Για την ολοκλήρωση της πρώτης εργαστηριακής άσκησης ζητάται να γίνει αποστολή ενός interrupt μετά από χρόνο 20ms με δύο τρόπους. Με βρόχο επανάληψης και μέσω ενός timer/counter με χρήση του prescaler. Ο μικροελεγκτής που χρησιμοποιήθηκε για την υλοποίηση είναι ο Atmega16A και η υλοποίηση και των δύο τρόπων έχει γίνει χρησιμοποιώντας γλώσσα Assembly. Η περιγραφή της υλοποίησης ακολουθεί παρακάτω:

1 Βρόχος επαναλήψεων

Αρχικά είναι απαραίτητο να οριστεί ο αριθμός των επαναλήψεων που καλείται να τρέξει το CPU του μικροελεγκτή ώστε να επιτύχει την ζητούμενη καθυστέρηση. Ο τύπος που υπολογίζει την ζητούμενη καθυστέρηση με την χρήση μίας επαναληπτικής διαδικασίας είναι ο εξής:

$$T_{delay} = \frac{1}{f_{clk}}(No_{reps} * L_{clk} - 1)$$

f_{clk} : Συχνότητα του ρολογιού του CPU

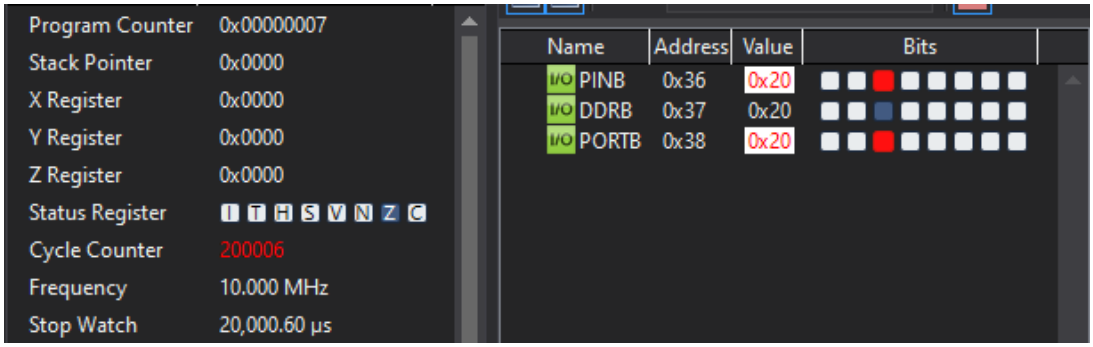
No_{reps} : Αριθμός επαναλήψεων των εντολών της διαδικασίας

L_{clk} : Αριθμός επαναλήψεων των εντολών της διαδικασίας

Το -1 οφείλεται στο γεγονός ότι για την έξοδο από την επαναλήπτική διαδικασία θα χρειαστεί μία εντολή διακλάδωσης η οποία σε περίπτωση ικανοποίησης της συνθήκης χρειάζεται έναν κύκλο λιγότερο από την αντίθετη περίπτωση. Οι εντολές του βρόχου επανάληψης που χρησιμοποιήθηκαν είναι οι sbiw και η brne(Σύνολο 4 κύκλοι). Μέσω των πράξεων προκύπτει ότι ο αριθμός επαναλήψεων είναι 50.000 για μία καθυστέρηση 20ms(Ρολόι CPU = 10MHz). Αναλογικά για καθυστέρηση 1s θα γίνει εκτέλεση περίπου

5.000.000 εντολών. Αναλυτικά αφού η εκτέλεση γίνεται σε 200.000 κύκλους και οι εντολές είναι 2 σημαίνει ότι πραγματοποιούνται 100.000 εντολές. Συνεπώς αφού εκτελούνται περίπου 100.000 εντολές για 20ms, αναλογικά θα εκτελούνται περίπου 5.000.000 εντολές για 1s

Μέσω της χρήσης των breakpoints παίρνουμε το παρακάτω αποτέλεσμα:
Σημαντική παρατήρηση ότι είναι ότι στο ρολόι δείχνει κάποιους παραπάνω κύκλους και



κατά συνέπεια και κάποια παραπάνω μs από το επιθυμητό. Αυτό οφείλεται σε εντολές που αρχικοποιούν τους καταχωρητές καθώς και την ενεργοποίηση του PortB για την εξαγωγή του interrupt.

2 Χρήση Timer/Counter και Prescaler

Για την υλοποίηση της καθυστέρησης χρησιμοποιήθηκε ο Timer0. Η επιλογή αυτή έγινε καθώς θεωρήθηκε ότι τα 8bit είναι αρκετά για την ολοκλήρωση της διαδικασίας. Για την χρήση του Timer0 (8 bit timer), γίνεται χρήση του παρακάτω τύπου για να βρεθεί ο κατάλληλος prescaler.

$$P_{val} = \frac{P_{clk}}{TOV_{clk} * Max\_Val}$$

P_{val}: Prescaler value
P_{clk}: Συχνότητα ρολογιού
TOV_{clk}: Overflows/sec
Max_Val: $2^{timerlen} - 1$

Με εφαρμογή του 255 ως Max_Val του 50 ως TOV_{clk} και της δεδομένης συχνότητας του ρολογιού προκύπτει $P_{val} = 784$. Συνεπώς, επιλέγεται η αμέσως επόμενη τιμή 1024. Με την λύση του ίδιου τύπου ως προς την μεταβλητή TOV_{clk} παρατηρείται ότι ο timer θα παρουσιάζει 0.7 overflows/s. Συνεπώς χρειάζεται αρχικοποίηση. Για να γίνει

σωστά η αρχικοποίηση υπολογίζεται η 'νέα' συχνότητα του ρολογιού με την εφαρμογή του prescaler και πολλαπλασιάζεται επί 256. Το αποτέλεσμα που προκύπτει είναι 26,2 ms. Συνεπώς με απλή μέθοδο των τριών προκύπτει ο αριθμός των κύκλων που χρείαζεται να παραληφθούν για να γίνει το overflow στην επιθυμητή καθυστέρηση(60). Σε επίπεδο κώδικα, θέτονται δύο directives μέσω της .org. Μία για το reset στο $000 (reset) και μία στο $012(timer_overflow) που είναι ο interrupt handler του timer0. Στο tag reset γίνεται αρχικά ορισμός του DDRB ώστε να χρησιμοποιηθεί ως έξοδος, αρχικοποιείται ο timer στην κατάλληλη τιμή, και μέσω εντολών ldi και out τίθεται ο prescaler και ενεργοποιείται ο Timer Overflow Flag. Για να μην υπάρχουν interrupts από άλλα units ενεργοποιείται ο Timer Interrupt Mask με τον ίδιο τρόπο. Όταν θα έρθει ο interrupt handler του overflow του timer μέσω της εντολής sbi στέλνουμε ένα εξωτερικό σήμα στο PORTB, σταματάται ο timer και το πρόγραμμα οδηγείται σε ατέρμονο loop για τους σκοπούς της άσκησης. Κάνοντας τον υπολογισμό των εντολών για 1s ακολουθείται η ίδια διαδικασία με την αναλογία στο βρόχο επανάληψης. Θεωρώντας ότι το πρόγραμμα θα επαναλαμβάνεται κάθε 20ms, οι εντολές αρχικοποίησης (11) θα επαναλαμβάνονται 50 φορές. Αυτό επιβεβαιώνεται καθώς 50*20ms =1s. Συνεπώς, αναλογικά σε 1s θα εκτελεστούν 11*50 = 550 εντολές.

Τα αποτελέσματα φαίνονται στην παρακάτω εικόνα:

Παρατηρείται ότι ο μετρητής μας είναι πάρα πολύ ακριβής καθώς τα παραπάνω μs που εμφανίζονται οφείλονται στις εντολές αρχικοποίησης του timer και των interrupts.

Συμπεράσματα:

Μέσω της άσκησης μπορεί να βγει το συμπέρασμα ότι μέσω της χρήσης βρόχου επανάληψης, η καθυστέρηση που απαιτείται μπορεί να επιτευχθεί με ακρίβεια αλλά υπάρχει το πρόβλημα κατανάλωσης των κύκλων του ρολογιού. Με άλλα λόγια ο microcontroller καταναλώνει κύκλους CPU περιμένοντας μέχρι να στείλει το interrupt χωρίς να έχει την δυνατότητα να τους χρησιμοποιήσει για να εκτελέσει κάποια άλλη διεργασία στη διάρκεια του delay. Αυτό το πρόβλημα αντιμετωπίζεται με την χρήση των timers μιας και όπως φάνηκε από τα αποτελέσματα, το interrupt έρχεται με μεγάλη ακρίβεια αφήνοντας

τον επεξεργαστή να χρησιμοποιήσει τους κύκλους σε άλλες διεργασίες όσο περιμένει τον timer να επιστρέψει το interrupt για να το διαχειριστεί κατάλληλα. Συνεπώς, η χρήση των timers για τέτοιου τύπου interrupts κρίνεται απαραίτητη για την σωστή διαχείρηση των πόρων του microcontroller. Αυτό επιβεβαιώνεται επιπλέον από την διαφορά του ποσού των εντολών, καθώς με την διαδικασία του βρόχου επανάληψης χρειάζονται περίπου 5 εκατομύρια εντολές, ποσό πολύ μεγαλύτερο από το ποσό των 550 εντολών που θα χρειαστούν με την χρήση των timers.