ΗΡΥ411- Ενσωματωμένα Συστήματα Μικροεπεξεργαστών

Εργαστήριο 2

LAB41145851 **15/10/2020**

Εμμανουήλ Πετράκος ΑΜ 2014030009

Για το δεύτερο εργαστήριο έχει κατασκευαστεί το πρόγραμμα οδήγησης μιας 7-segment οθόνης οκτώ ψηφίων. Ως είσοδος του προγράμματος θεωρείται ένα κομμάτι μνήμης SRAM που περιέχει τα δεδομένα προς εμφάνιση. Η έξοδος του συστήματος είναι το σήμα 7-segment (A-G,DP) στο PORTA και το σήμα ενεργοποίησης των ψηφίων (AN7-AN0) στο PORTC.

Επεξήγηση προσέγγισης

Το πρόγραμμα ξεκινάει με τις απαραίτητες αρχικοποιήσεις. Όπως και στο προηγούμενο εργαστήριο, αρχικοποιείται ο stack pointer για να μπορούν να χρησιμοποιηθούν interrupts και υπορουτίνες, καθώς και ο TIMER0 ρυθμισμένος στα 2ms. Με αυτή την ρύθμιση, η συχνότητα ανανέωσης είναι $1s / (2ms * 8 ψηφία) \approx 62,5 Hz/ψηφίο$, περίπου δύο φορές μεγαλύτερη από το ελάχιστο όριο. Οι θύρες A και C ορίζονται ως έξοδοι και αρχικοποιούνται με τέτοιο τρόπο ώστε να μην ενεργοποιηθεί ακόμα κάποιο LED και να ξεκινήσει η λειτουργία από το AN0. Το PORTC λειτουργεί σαν ένας μετρητής δακτυλίου, έχοντας ενεργοποιημένο μόνο ένα bit που αντιστοιχεί στο ψηφίο που εμφανίζεται.

Το πρόγραμμα χρησιμοποιεί 18 bytes στην SRAM, 8 για την είσοδο του και 10 για τις κωδικοποιήσεις 7_segment. Χρησιμοποιώντας την ρουτίνα *write_data* αποθηκεύεται η είσοδος, ξεκινώντας από την διεύθυνση 0x0060. Τα δεδομένα του ψηφίου 0 αποθηκεύονται στην πρώτη θέση, του ψηφίου 1 στην δεύτερη θέση κλπ. Με αυτό τον τρόπο, ο δείκτης μιας εξόδου μπορεί να χρησιμοποιηθεί ως offset πάνω στην αρχική διεύθυνση για να γίνει πρόσβαση στα αντίστοιχα δεδομένα. Η αποθήκευση των κωδικοποιήσεων γίνεται μέσω της ρουτίνας *write_7_segments* αμέσως μετά τα δεδομένα εισόδου, δηλαδή ξεκινώντας από την διεύθυνση 0x0068. Ακολουθείται παρόμοια λογική με πριν, η κωδικοποίηση του αριθμού 0 αποθηκεύεται στην πρώτη θέση, του αριθμού 1 στη δεύτερη κλπ. Σαν αποτέλεσμα, ο ίδιος ο αριθμός λειτουργεί ως offset κατά την πρόσβαση στην κωδικοποίηση του. Γενικά, η αποθήκευση των δεδομένων έχει γίνει με γνώμονα την ευκολότερη αναζήτηση και πρόσβαση τους.

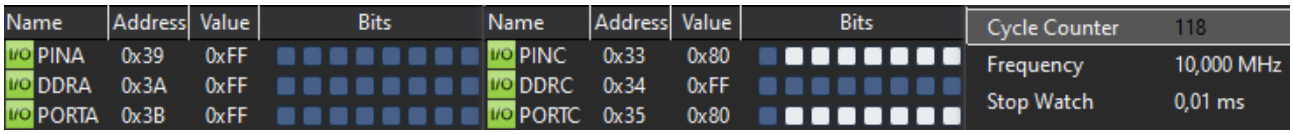
$$Υπολογισμός\ δικτών, όπου\ x\ το\ ενεργοποιημένο\ ψηφίο$$
$$(DATA.x) = 0x0060 + x$$
$$(7\,segment.y) = 0x0068 + DATA.x$$

Τέλος, ενεργοποιούνται τα interrupts και το πρόγραμμα μπαίνει σε ένα ατέρμον βρόχο.

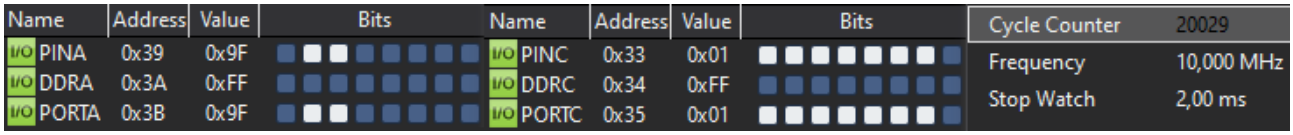
Η κύρια λειτουργικότητα του προγράμματος υλοποιείται στην ρουτίνα εξυπηρέτησης του interrupt. Ξεκινώντας, απενεργοποιούνται όλα τα LED για την αποφυγή εμφάνισης σκουπιδιών κατά την αλλαγή ψηφίου και δεδομένων. Κάνοντας τον μετρητή δακτυλίου στο PORTC κυκλικό shift προς τα αριστερά, απενεργοποιείται το τρέχων ψηφίο και ενεργοποιείται το επόμενο. Επίσης, η νέα τιμή του δίνει αρκετή πληροφορία για το πιο byte πρέπει να διαβαστεί από την μνήμη, καθώς μετατρέποντας τον αριθμό δακτυλίου σε δυαδική μορφή μπορεί να γίνει η πρόσβαση στα αντίστοιχα δεδομένα με τον τρόπο που αναφέρθηκε παραπάνω. Το επόμενο βήμα είναι η μετατροπή τους σε 7 segment μορφή. Γίνεται έλεγχος εγκυρότητας των δεδομένων, αν δεν είναι ένα από τα αποδεκτά ψηφία (0-9) απενεργοποιούνται όλα τα LED, αλλιώς διαβάζεται από την μνήμη η κατάλληλη κωδικοποίηση και κατευθύνεται στο PORTA.

Πειραματική Διαδικασία

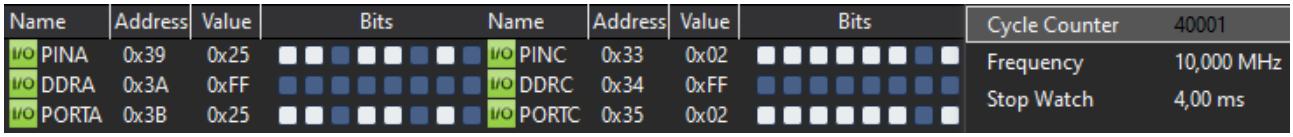
Αποθηκεύοντας στην μνήμη τους αριθμούς 1 έως 8, η οθόνη πρέπει να δείξει 87654321. Επειδή δεν υπάρχει διαθέσιμο υλικό, η επαλήθευση της λειτουργικότητας του προγράμματος γίνεται μέσω του simulation του Atmel Studio 7. Παρακολουθώντας τα PORTA και PORTC στον simulator είναι δυνατόν να αποφανθούμε τι θα εμφάνιζε η οθόνη την συγκεκριμένη χρονική στιγμή. Αυτό γίνεται με ένα breakpoint στο τέλος της ρουτίνας εξυπηρέτησης του interrupt και τα παράθυρα I/O και Processor Status.



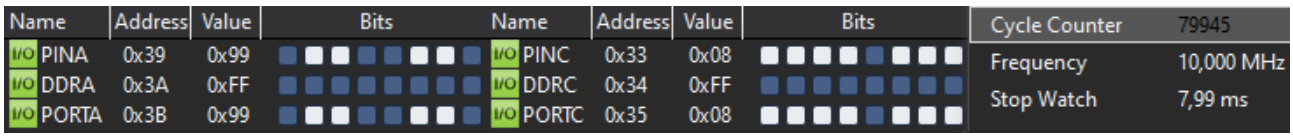
Εκκίνηση προγράμματος, PORTA = 0b11111111 (τίποτα)



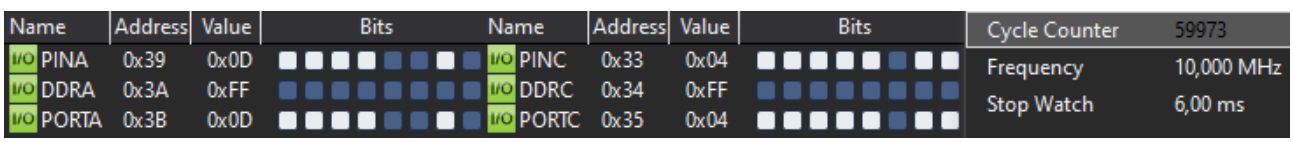
1ο interrupt: PORTA = 0b10011111 (1), PORTC = AN0



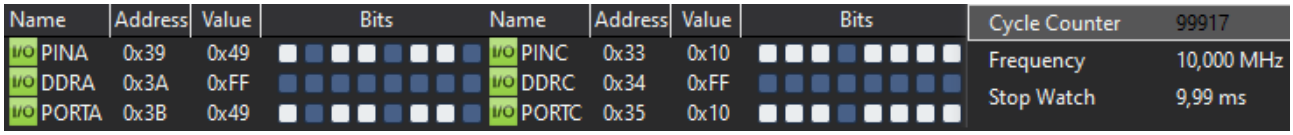
2ο interrupt: PORTA = 0b00100101 (2), PORTC = AN1



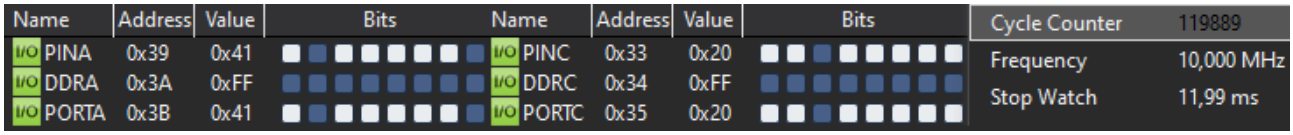
4ο interrupt: PORTA = 0b10011001 (4), PORTC = AN3



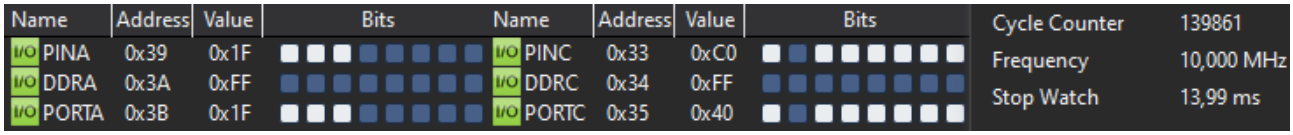
3ο interrupt: PORTA = 0b00001101 (3), PORTC = AN2



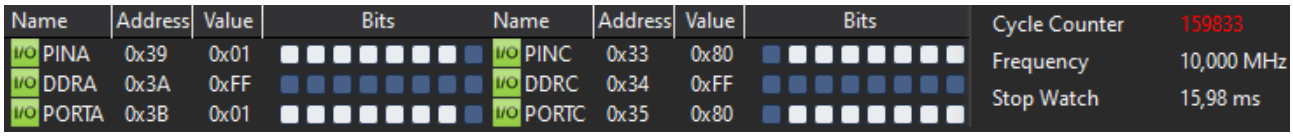
5ο interrupt: PORTA = 0b01001001 (5), PORTC = AN4



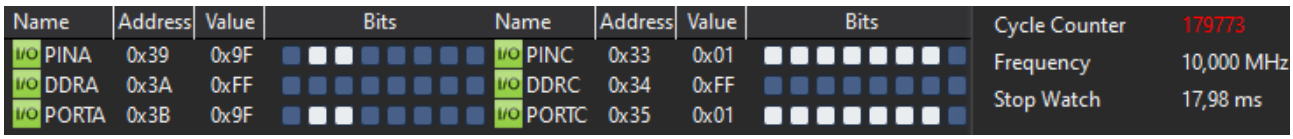
6ο interrupt: PORTA = 0b01000001 (6), PORTC = AN5



7ο interrupt: PORTA = 0b00011111 (7), PORTC = AN6

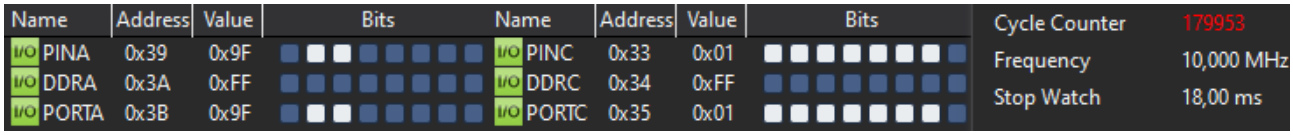


8ο interrupt: PORTA = 0b00000001 (8), PORTC = AN7

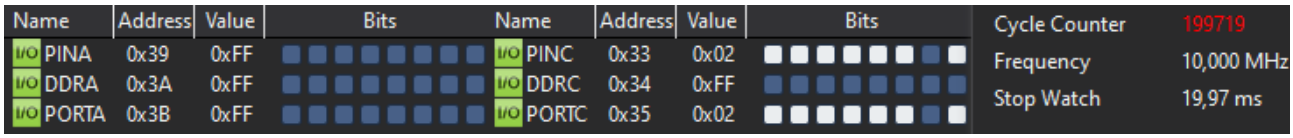


9ο interrupt: PORTA = 0b10011111 (1), PORTC = AN0

Πέρα από την σωστή αλλαγή των εξόδων, πρέπει να παραμένουν σταθερές όσο το πρόγραμμα είναι στον ατέρμον βρόχο και να μην εμφανίζονται σκουπίδια όσο εξυπηρετείται το interrupt.

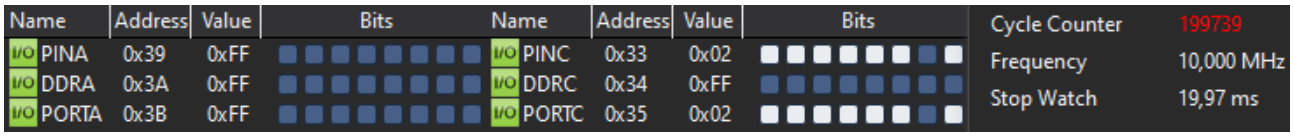


Ατέρμον βρόχος: PORTA = 0b10011111 (1), PORTC = AN0



Μέσα στο επόμενο Interrput: PORTA = 0b11111111 (τίποτα)

Τέλος, εμφανίζεται η έξοδος όταν η είσοδος έχει μη αποδεκτά δεδομένα.



Θέτοντας στην θέση 2 την τιμή 10: PORTA = 0b11111111 (τίποτα)

Ανάλυση & Παρατηρήσεις

Μεταξύ δύο διαδοχικών αλλαγών παρέρχονται ~20000 κύκλοι ρολογιού, όπου ~60 από αυτούς χρειάζονται για την εξυπηρέτηση του interrupt. Δηλαδή, ο driver απαιτεί ~0.3% του υπολογιστικού χρόνου του επεξεργαστή.

Για το παρόν εργαστήριο τα δεδομένα προς εμφάνιση ορίζονται στην ρουτίνα *write_data* κατά την εκκίνηση του προγράμματος και μένουν σταθερά, κάτι που πρέπει να αλλάξει στα επόμενα εργαστήρια για να είναι ο driver χρήσιμος. Πέρα από αυτό, το πρόγραμμα είναι ολοκληρωμένο.

Πηγές

7 Segment Encoding
www.elprocus.com/bcd-to-seven-segment-display-decoder-theory/

Using SRAM in AVR
www.avr-asm-tutorial.net/avr_en/beginner/SRAM.html

AVR Instruction Set Manual
http://ww1.microchip.com/downloads/en/devicedoc/atmel-0856-avr-instruction-set-manual.pdf