

# ΗΡΥ411- Ενσωματωμένα Συστήματα Μικροεπεξεργαστών

## Εργαστήριο 2

LAB41145851

15/10/2020

Εμμανουήλ Πετράκος AM 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

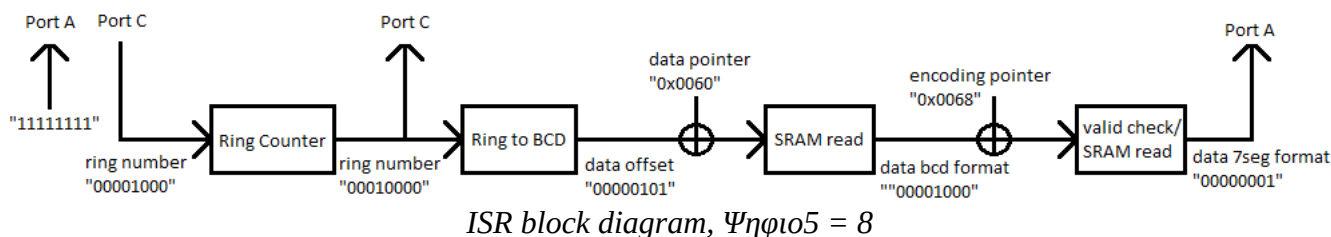
Το πρόγραμμα ξεκινάει με τις απαραίτητες αρχικοποιήσεις. Όπως και στο προηγούμενο εργαστήριο, αρχικοποιείται ο stack pointer για να μπορούν να χρησιμοποιηθούν interrupts και υπορουτίνες, καθώς και ο TIMER0 ρυθμισμένος στα 2ms. Με αυτή την ρύθμιση, η συχνότητα ανανέωσης είναι  $1s / (2ms * 8\text{ψηφία}) \approx 62,5\text{Hz/ψηφίο}$ , 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.

$$\begin{aligned} &\text{Υπολογισμός δικτών, όπου } x \text{ το ενεργοποιημένο ψηφίο} \\ &(DATA.x) = 0x0060 + x \\ &(7\_segment.y) = 0x0068 + DATA.x \end{aligned}$$

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 εμφανίζε η οθόνη την συγκεκριμένη χρονική στιγμή. Αυτό γίνεται με ένα breakpoint στο τέλος της ISR και τα παράθυρα I/O και Processor Status.

Name	Address	Value	Bits	Name	Address	Value	Bits	Cycle Counter	118
I/O PINA	0x39	0xFF	11111111	I/O PINC	0x33	0x80	10000000	Frequency	10,000 MHz
I/O DDRA	0x3A	0xFF	11111111	I/O DDRC	0x34	0xFF	11111111	Stop Watch	0,01 ms
I/O PORTA	0x3B	0xFF	11111111	I/O PORTC	0x35	0x80	10000000		

*Εκκίνηση προγράμματος, PORTA = 0b11111111 (τίποτα)*

Name	Address	Value	Bits	Name	Address	Value	Bits	Cycle Counter	20029
I/O PINA	0x39	0x9F	10011111	I/O PINC	0x33	0x01	00000001	Frequency	10,000 MHz
I/O DDRA	0x3A	0xFF	11111111	I/O DDRC	0x34	0xFF	11111111	Stop Watch	2,00 ms
I/O PORTA	0x3B	0x9F	10011111	I/O PORTC	0x35	0x01	00000001		

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

Name	Address	Value	Bits	Name	Address	Value	Bits	Cycle Counter	40001
I/O PINA	0x39	0x25	00101001	I/O PINC	0x33	0x02	00000010	Frequency	10,000 MHz
I/O DDRA	0x3A	0xFF	11111111	I/O DDRC	0x34	0xFF	11111111	Stop Watch	4,00 ms
I/O PORTA	0x3B	0x25	00101001	I/O PORTC	0x35	0x02	00000010		

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

Name	Address	Value	Bits	Name	Address	Value	Bits	Cycle Counter	59973
I/O PINA	0x39	0x0D	00001101	I/O PINC	0x33	0x04	00000100	Frequency	10,000 MHz
I/O DDRA	0x3A	0xFF	11111111	I/O DDRC	0x34	0xFF	11111111	Stop Watch	6,00 ms
I/O PORTA	0x3B	0x0D	00001101	I/O PORTC	0x35	0x04	00000100		

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

Name	Address	Value	Bits	Name	Address	Value	Bits	Cycle Counter	79945
I/O PINA	0x39	0x99		I/O PINC	0x33	0x08		Frequency	10,000 MHz
I/O DDRA	0x3A	0xFF		I/O DDRC	0x34	0xFF		Stop Watch	7,99 ms
I/O PORTA	0x3B	0x99		I/O PORTC	0x35	0x08			

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

Name	Address	Value	Bits	Name	Address	Value	Bits	Cycle Counter	99917
I/O PINA	0x39	0x49		I/O PINC	0x33	0x10		Frequency	10,000 MHz
I/O DDRA	0x3A	0xFF		I/O DDRC	0x34	0xFF		Stop Watch	9,99 ms
I/O PORTA	0x3B	0x49		I/O PORTC	0x35	0x10			

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

Name	Address	Value	Bits	Name	Address	Value	Bits	Cycle Counter	119889
I/O PINA	0x39	0x41		I/O PINC	0x33	0x20		Frequency	10,000 MHz
I/O DDRA	0x3A	0xFF		I/O DDRC	0x34	0xFF		Stop Watch	11,99 ms
I/O PORTA	0x3B	0x41		I/O PORTC	0x35	0x20			

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

Name	Address	Value	Bits	Name	Address	Value	Bits	Cycle Counter	139861
I/O PINA	0x39	0x1F		I/O PINC	0x33	0xC0		Frequency	10,000 MHz
I/O DDRA	0x3A	0xFF		I/O DDRC	0x34	0xFF		Stop Watch	13,99 ms
I/O PORTA	0x3B	0x1F		I/O PORTC	0x35	0x40			

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

Name	Address	Value	Bits	Name	Address	Value	Bits	Cycle Counter	159833
I/O PINA	0x39	0x01		I/O PINC	0x33	0x80		Frequency	10,000 MHz
I/O DDRA	0x3A	0xFF		I/O DDRC	0x34	0xFF		Stop Watch	15,98 ms
I/O PORTA	0x3B	0x01		I/O PORTC	0x35	0x80			

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

Name	Address	Value	Bits	Name	Address	Value	Bits	Cycle Counter	179773
I/O PINA	0x39	0x9F		I/O PINC	0x33	0x01		Frequency	10,000 MHz
I/O DDRA	0x3A	0xFF		I/O DDRC	0x34	0xFF		Stop Watch	17,98 ms
I/O PORTA	0x3B	0x9F		I/O PORTC	0x35	0x01			

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.

Name	Address	Value	Bits	Name	Address	Value	Bits	Cycle Counter	179953
I/O PINA	0x39	0x9F		I/O PINC	0x33	0x01		Frequency	10,000 MHz
I/O DDRA	0x3A	0xFF		I/O DDRC	0x34	0xFF		Stop Watch	18,00 ms
I/O PORTA	0x3B	0x9F		I/O PORTC	0x35	0x01			

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

Name	Address	Value	Bits	Name	Address	Value	Bits	Cycle Counter	199719
I/O PINA	0x39	0xFF		I/O PINC	0x33	0x02		Frequency	10,000 MHz
I/O DDRA	0x3A	0xFF		I/O DDRC	0x34	0xFF		Stop Watch	19,97 ms
I/O PORTA	0x3B	0xFF		I/O PORTC	0x35	0x02			

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

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

Name	Address	Value	Bits	Name	Address	Value	Bits	Cycle Counter	199739
I/O PINA	0x39	0xFF	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	I/O PINC	0x33	0x02	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	Frequency	10,000 MHz
I/O DDRA	0x3A	0xFF	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	I/O DDRC	0x34	0xFF	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	Stop Watch	19,97 ms
I/O PORTA	0x3B	0xFF	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	I/O PORTC	0x35	0x02	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>		

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/](http://www.elprocus.com/bcd-to-seven-segment-display-decoder-theory/)

Using SRAM in AVR

[www.avr-asm-tutorial.net/avr\\_en/beginner/SRAM.html](http://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>