HPY411- Ενσωματωμένα Συστήματα Μικροεπεξεργαστών

Εργαστήριο 9

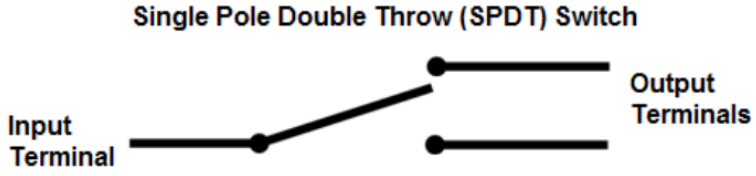
LAB41145851 **14/12/2020**

Εμμανουήλ Πετράκος ΑΜ 2014030009

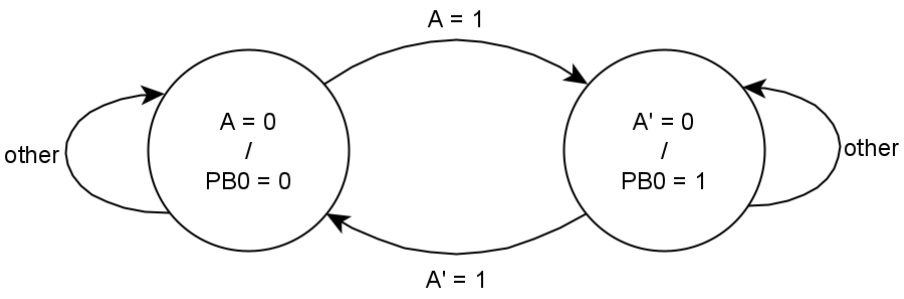
Στο 9ο εργαστήριο υλοποιείται κώδικας για την αφαίρεση αναπηδήσεων από ένα διακόπτη SPDT. Έχουν κατασκευαστεί δύο λύσεις, μια με polling βασισμένο σε timer και μία με εξωτερικά interrupts. Βρίσκονται στα αρχεία polling/main.c και interrupt/main.c αντίστοιχα.

Επεξήγηση προσέγγισης

Ένας SPDT διακόπτης αποτελείται από τρία ηλεκτρόδια, ένα κινητό (pole) και δύο σταθερά (throw). Όταν το pole αγγίζει ένα throw, στην αντίστοιχη είσοδο του μικροελεγκτή εμφανίζεται λογικό 0. Το pole βρίσκεται σε ηρεμία πάνω στο ένα throw, μέχρι να γυρίσει τον διακόπτη ο χρήστης όπου μετακινείται προς το άλλο. Τότε, για ένα μικρό διάστημα δεν αγγίζει κανένα και αφού φτάσει στο δεύτερο throw, λόγο της ελαστικότητας των υλικών, αναπηδάει και εμφανίζονται και άλλα τέτοια διαστήματα. Τότε εμφανίζεται το φαινόμενο όπου για ένα μικρό χρονικό διάστημα και οι δύο είσοδοι του μικροελεγκτή έχουν την τιμή 1 καθώς και η είσοδος που αναπηδάει το pole αλλάζει πολλές φορές τιμή. Αυτό είναι ανεπιθύμητο και πρέπει να αφαιρεθεί.

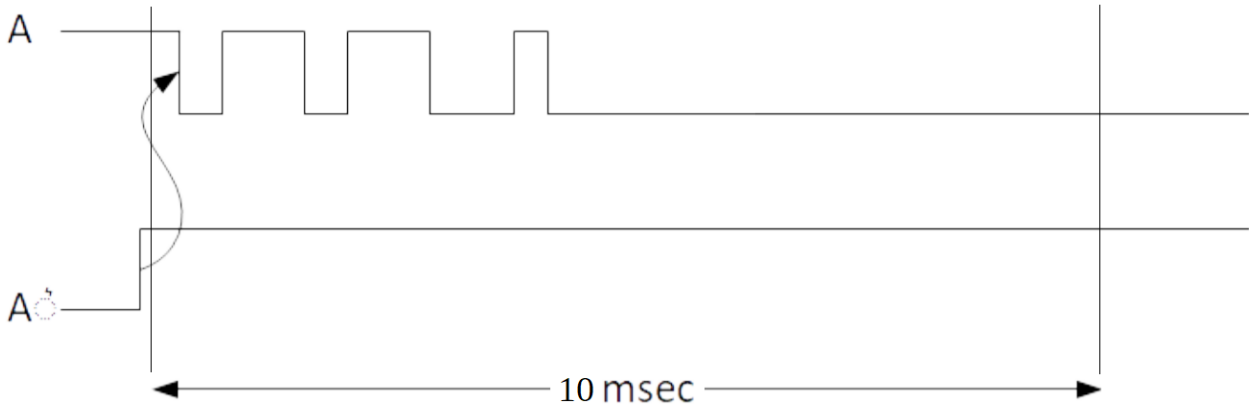


Η λύση με polling βασίζεται στην παρατήρηση ότι στο throw από όπου έφυγε το pole δεν εμφανίζεται το φαινόμενο της αναπήδησης. Παρακολουθώντας μόνο τις αλλαγές πάνω σε αυτό, το σύστημα μπορεί να αποφανθεί για την κατάσταση του διακόπτη. Τότε είναι αναγκαία η χρήση μνήμης για να θυμάται την προηγούμενη κατάσταση των throws και παράγεται η παρακάτω fsm. Τέλος, στην μνήμη αποθηκεύεται η κατάσταση που έπρεπε να είναι, ώστε να μην επηρεάσει μια αναπήδηση την συμπεριφορά του συστήματος κατά την επόμενη δειγματοληψία.



Polling FSM

Επίσης, αν η περίοδος δειγματοληψίας είναι μεγαλύτερη από τον σύμφωνα με τις προδιαγραφές χρόνο ταλάντωσης, τότε αποφεύγονται λάθος αλλαγές λόγο των αναπηδήσεων στη νέα κατάσταση. Δηλαδή, ακόμα και αν η δειγματοληψία γίνει απευθείας μετά την αλλαγή, το σύστημα θα έχει φτάσει σε ηρεμία όταν παρθεί το επόμενο δείγμα και μελετάται η επόμενη κατάσταση. Για αυτό η περίοδος δειγματοληψίας τίθεται στα 10 ms.



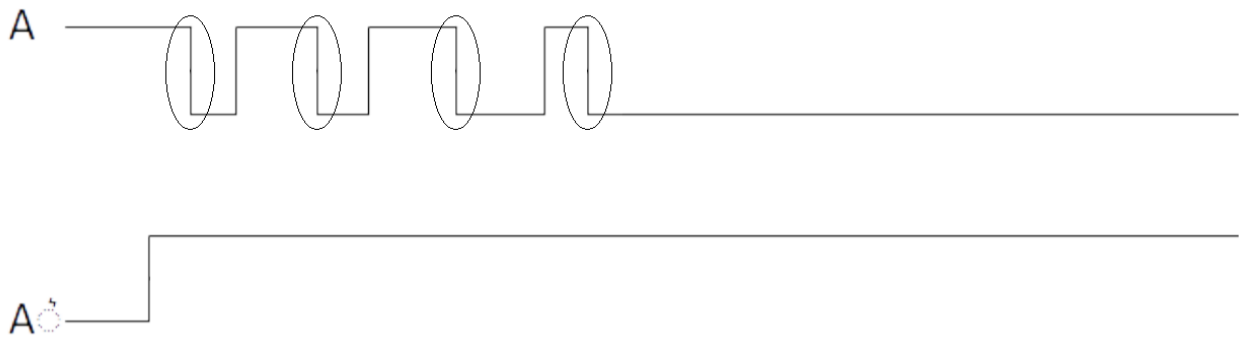
Δειγματοληψία αμέσως μετά από μία αλλαγή. Οι κάθετες γραμμές είναι δύο συνεχόμενα δείγματα.

Η υλοποίηση των παραπάνω είναι αρκετά απλή και γίνεται στο αρχείο main.c του project polling. Στη συνάρτηση `init_polling_driver()` αρχικοποιείται το σύστημα σύμφωνα με τον παρακάτω πίνακα και ρουτίνα αντιμετώπισης interrupts του μετρητή 0, `ISR( TIMER0_COMP_vect )`, υλοποιείται η λειτουργικότητα της παραπάνω FSM.

Διεύθυνση	Τιμή	Ρύθμιση
DDRA	|= 0x03	Χρήση Port A bit 0 & 1 ως είσοδος
DDRB	|= 0x01	Χρήση Port B bit 0 ως έξοδος
OCR0	97	10ms Period @ 10MHz F_CPU
TCCR0		
CS02:0	101	$clk_{I/O}$/1024 (From prescaler)
WGM01:0	10	Compare mode: Clear Counter on match
TIMSK		
OCIE0	1	Ενεργοποίηση interrupt Compare Match

Η λύση με interrupts εκμεταλλεύεται την ίδια παρατήρηση, δηλαδή ότι το φαινόμενο της αναπήδησης εμφανίζεται μόνο στον ένα ακροδέκτη. Πιο συγκεκριμένα, σε κάθε μετακίνηση του διακόπτη, μόνο στο τελικό throw θα εμφανιστούν μεταβολές από 1 σε 0. Δηλαδή το throw που εμφάνισε τελευταίο 0, είναι αυτό όπου βρίσκεται το pole.

Τότε, έχοντας τα interrupts σε falling edge mode αυτά ενεργοποιούνται τις επιθυμητές στιγμές και το μόνο που χρειάζεται να γίνει είναι να τεθεί το αντίστοιχο throw ως το ενεργό (μια ISR για κάθε throw). Στην παρακάτω εικόνα φαίνεται πότε ενεργοποιούνται τα interrupts σε μια μετάβαση. Όσες φορές και να κληθεί μια ISR, το αποτέλεσμα θα είναι το ίδιο, καθώς ξανακάνει την ίδια ανάθεση.



Ενεργοποίηση Interrupts σε falling edge mode

Η αρχικοποίηση του συστήματος περιγράφεται στον παρακάτω πίνακα.

Διεύθυνση	Τιμή	Ρύθμιση
DDRB	|= 0x01	Χρήση Port B bit 0 ως έξοδος
MCUCR		
ISC11:0	10	External Int1 at Falling Edge mode
ISC01:0	10	External Int0 at Falling Edge mode
GIGR		
INT1	1	Enable external Int1
INT0	1	Enable external Int0

Πειραματική Διαδικασία

Η επιβεβαίωση της σωστής λειτουργίας του συστήματος γίνεται προσομοιώνοντας έναν SPDT διακόπτη στο αρχείο SPDT.stim. Στη περίπτωση της λύσης με polling χρησιμοποιείται το PINA. Στον παρακάτω πίνακα φαίνεται αναλυτικά η λειτουργία του.

Time (ms)	A1	A0	
0	0	1	Αρχική κατάσταση διακόπτη
10.1	1	1	Μετακίνηση διακόπτη, το pole δεν αγγίζει κανένα throw.
10.11	1	0	Το pole χτυπάει το δεύτερο throw πρώτη φορά.
10.12	1	1	Πολλές αναπηδήσεις στο throw που χτυπήθηκε, σε τυχαία χρονικά διαστήματα μεταβλητού μεγέθους.
10.22	1	0	
11.22	1	1	
11.42	1	0	
11.72	1	1	
11.82	1	0	
12.82	1	1	
13.02	1	0	
13.17	1	1	
13.37	1	0	Ο διακόπτης φτάνει σε ηρεμία.
28.57	1	1	Μετακίνηση διακόπτη προς το αρχικό throw.
28.58	0	1	Χτυπάει το πρώτο throw πρώτη φορά.

28.68	1	1	Πολλές αναπηδήσεις στο throw που χτυπήθηκε, σε τυχαία χρονικά διαστήματα μεταβλητού μεγέθους.
31.68	0	1	
31.78	1	1	
32.11	0	1	

Χρησιμοποιώντας ένα breakpoint στο τέλος της ISR του timer0, φαίνεται η κατάσταση στις εισόδους και την έξοδο μετά από κάθε δειγματοληψία. Το .stim αρχείο είναι κατασκευασμένο έτσι ώστε να γίνει μια δειγματοληψία πάνω σε αναπήδηση και να φανεί ότι το σύστημα αντιδράει ορθά.

Time (ms)	A1	A0	B	
0	0	1	1	Η έξοδος αρχικοποιείται κατά την εκκίνηση σύμφωνα με τις εισόδους.
10.04	0	1	1	1ο δείγμα. Καμία αλλαγή στην είσοδο που είναι 0 (A1), καμία αλλαγή στην έξοδο.
20.07	1	0	0	2ο δείγμα. A1: 0→1. Αλλαγή εξόδου
30.11	1	1	1	3ο δείγμα. A0: 0→1. Η δειγματοληψία γίνεται εν μέσω αναπήδησης αλλά η κατάσταση της A1 είναι αδιάφορη.
40.14	0	1	1	4ο δείγμα. Η A1: 1→ 0. Δεν επιφέρει καμία αλλαγή στην έξοδο καθώς η προηγούμενη τιμή προερχόταν από αναπήδηση. Στο προηγούμενο δείγμα η μνήμη για το A1 είχε πάρει την τιμή 0.

Φαίνεται ότι η τιμή του B αλλάζει όσες φορές μετακινείται ο διακόπτης, και ότι τα rebounce δεν επηρεάζουν την λειτουργία του συστήματος.

Για τον έλεγχο του συστήματος με interrupts χρησιμοποιείται το ίδιο .stim αρχείο με μόνη διαφορά ότι οι οι είσοδοι A1 και A0 αντικαθιστώνται με τις D3 και D2. Χρησιμοποιώντας breakpoints στο τέλος των ISR των εξωτερικών interrupts, φαίνεται η κατάσταση στις εισόδους και την έξοδο κάθε φορά που το σύστημα απασχολείται με τον διακόπτη.

Time (ms)	D3	D2	B	
0	0	1	1	Η έξοδος αρχικοποιείται κατά την εκκίνηση σύμφωνα με τις εισόδους.
10.11	1	0	0	D2: 1→ 0 λόγο μετακίνησης του διακόπτη. B = 0.
10.22	1	0	0	D2: 1→ 0 λόγο αναπήδησης. Κάνουν το B και πάλι 0, αλλά είχε ήδη αυτήν την τιμή και το σύστημα δεν αντιλαμβάνεται κάποια αλλαγή.
11.42	1	0	0	
11.82	1	0	0	
13.02	1	0	0	
13.37	1	0	0	
28.58	0	1	1	D3: 1→ 0 λόγο μετακίνησης του διακόπτη. B = 0.
31.68	0	1	1	D3: 1→ 0 λόγο αναπήδησης. Κάνουν το B και πάλι 1, αλλά είχε ήδη αυτήν την τιμή και το σύστημα δεν αντιλαμβάνεται κάποια αλλαγή.
32.11	0	1	1	

Παρατηρείται ότι όσες φορές και να κληθούν οι ISR, η τιμή του B αλλάζει όσες φορές μετακινείται ο διακόπτης. Οι κλήσεις τους λόγο των αναπηδήσεων δεν επηρεάζει την έξοδο.

Και στις δύο λύσεις, στο εξαγόμενο αρχείο lab.log εμφανίζονται τρεις αλλαγές του διακόπτη, μία από την αρχικοποίηση και δύο από την μεταβολή του διακόπτη. Αυτό που διαφέρει είναι ο χρόνος μεταξύ των αλλαγών.

Ανάλυση & Παρατηρήσεις

Έχοντας δύο λύσεις για το ίδιο πρόβλημα γεννάται το ερώτημα ποια είναι η καλύτερη. Εφόσον η λειτουργικότητα και των δύο είναι εντός προδιαγραφών, το μέτρο σύγκρισης που απομένει είναι η κατανάλωση πόρων. Στο σύστημα AVR όπου υλοποιούνται, μπορούν να κατηγοριοποιηθούν ως οι περιφερειακοί πόροι και η κατανάλωση του υπολογιστικού χρόνου του επεξεργαστή.

Για την λύση με polling οι απαιτήσεις σε περιφερειακούς πόρους είναι:

- 2 οποιεσδήποτε θύρες για την είσοδο.
- 1 θύρα για την έξοδο.
- Ένας χρονομετρητής.
- 2 θέσεις μνήμης RAM.

Οι αντίστοιχες απαιτήσεις για την λύση με εξωτερικά interrupts είναι:

- Οι δύο θύρες που αντιστοιχούν στα εξωτερικά interrupt.
- 1 θύρα για την έξοδο.
- Η μονάδα εξωτερικών interrupts.

Ο υπολογισμός της κατανάλωσης επεξεργαστικών πόρων είναι πιο σύνθετος καθώς οι προδιαγραφές δεν δίνουν κάποια μέση η χειρότερη χρήση του συστήματος. Για αυτό πρέπει να γίνουν μερικές υποθέσεις και παραδοχές. Η πρώτη είναι κάθε πότε αλλάζει θέση ο διακόπτης. Σύμφωνα με έρευνες, η μέγιστη ταχύτητα που μπορεί ένας μέσος άνθρωπος να πατήσει ένα κουμπί είναι περίπου 100 ms[1]. Αυτός ο χρόνος μπορεί να χρησιμοποιηθεί ως η χειρότερη περίπτωση. Η δεύτερη υπόθεση που πρέπει να γίνει είναι πόσες αναπηδήσεις θα γίνουν με κάθε αλλαγή του διακόπτη. Αυθαίρετα υποθέτεται ότι είναι κατά μέσο όρο στις 10 για διευκόλυνση των υπολογισμών.

Στην περίπτωση του polling, η κατανάλωση της ISR του timer δεν είναι σταθερή, καθώς εξαρτάται από τον αν εντόπισε αλλαγή ή όχι. Στη πρώτη περίπτωση χρειάζεται 51 κύκλους ενώ στην δεύτερη 45. Μπορεί να υπολογιστεί πόσες φορές θα κληθεί ανά πάτημα κουμπιού από την παρακάτω σχέση.

$$\frac{t_{worst\ case}}{T_{I/O\ clock}} = \frac{100\,ms}{10\,ms} = 10 \quad \text{κλήσεις, όπου στην μία έγινε αλλαγή.}$$

Οι συνολική απαίτηση σε κύκλους ανά αλλαγή διακόπτη μπορεί να υπολογιστεί ως:

$$f_{no\ change} \cdot c_{no\ change} + f_{change} \cdot c_{change} = 9 \cdot 45 + 1 \cdot 51 = 456 \quad \text{κύκλους ανά αλλαγή διακόπτη.}$$

Η κατανάλωσης της λύσης με interrupts εξαρτάται από συχνότητα αλλαγής του διακόπτη και από τον αριθμό των αναπηδήσεων σε κάθε αλλαγή. Εφόσον γίνεται 1 αλλαγή κάθε 100ms και για κάθε μια δημιουργούνται 10 rebounce, γίνονται 11 κλήσεις της ISR ανά 100ms. Η κάθε κλήση κοστίζει 21 κύκλους και εύκολα υπολογίζεται ότι χρειάζονται 231 κύκλοι ανά 100ms.

Στο παρακάτω πίνακα συγκρίνεται η κατανάλωση των λύσεων για διαφορετικές συχνότητες ρολογιού.

		Κατανάλωση @ f_{cpu} 1 MHz		Κατανάλωση @ f_{cpu} 10 MHz	
Λύση	Κύκλοι / 100msec	t/100msec	%	t/100msec	%
Polling	456	456μsec	0.456	45.6μsec	0.0456
Interrupt	231	231μsec	0.231	23.1μsec	0.0231

Και στις δύο λύσεις, η κατανάλωση επεξεργαστικής δύναμης είναι από ελάχιστη έως αμελητέα. Στις περισσότερες εφαρμογές η επιλογή θα γίνει βάση των διαθέσιμων περιφερειακών πόρων, εκτός από εξαίρετες περιπτώσεις όπου η απελευθέρωση του 0.2% του επεξεργαστή είναι αναγκαία.

Οι παραπάνω λύσεις εκμεταλλεύονται ότι υπάρχουν δύο είσοδοι και πλεονασμός πληροφορίας για την κατάσταση του διακόπτη SPDT. Σε περίπτωση που ο διακόπτης αντικατασταθεί με έναν SPST και υπάρχει μια διαθέσιμη είσοδος, δεν λειτουργούν ορθά.

Επειδή η υλοποίηση του εργαστηρίου διαφέρει από την προτεινόμενη της εκφώνησης, στην αναφορά δίνεται μεγαλύτερο βάρος στην απόδειξη της ορθής λειτουργίας της.

Πηγές

[1]Pressing a button is more challenging than it appears
https://www.sciencedaily.com/releases/2018/03/180320100852.htm#:~:text=One%20problem%20the%20brain%20must,too%20fast%20for%20correcting%20movement.

Switch Contact bounce
https://en.wikipedia.org/wiki/Switch#Contact_bounce

SWITCH BOUNCE / DEBOUNCE TUTORIAL
https://www.logiswitch.net/switch-bounce-debounce-tutorial