ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΤΕΧΝΟΛΟΓΙΑΣ & ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ

ΕΡΓΑΣΤΗΡΙΟ ΠΡΟΗΓΜΕΝΩΝ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ

4Η ΑΣΚΗΣΗ

19/05/2022

ΓΕΩΡΓΙΟΣ ΚΟΝΤΟΓΙΑΝΝΗΣ - 1070908

ΧΑΡΑΛΑΜΠΟΣ ΠΑΠΠΑΣ - 1053359

Εννοποιημένος Κώδικας (1-3 ερωτήματα)

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

int pass_counter = 0;
int pass_input = 0;
int timer_after_pass = 0;
int tries = 0;
int is_robbed = 0;
int is_robbed_timer = 0;
int pulses = 0;

int main(void){

	// led 0 is output
	PORTD.DIR |= PIN0_bm;
	PORTD.OUT |= PIN0_bm;

	// set switches
	PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
	PORTF.PIN6CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;

	while(1){
		enable_alarm();

		ADC_init();

		disable_alarm();
	}
}

void enable_alarm()
{
	// make sure pass_input = 0
	pass_input = 0;
	sei();
	while(!pass_input){
		;
	}
	//pass_input = 0;
	//Timer set up
	TCA0.SINGLE.CNT = 0; //Clear counter
	TCA0.SINGLE.CTRLB = 0; //Normal Mode
	TCA0.SINGLE.CMP0 = 5; //Stop turning when this value is reached
	TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV1024_gc;
	TCA0.SINGLE.CTRLA |= 1;
	TCA0.SINGLE.INTCTRL = TCA_SINGLE_CMP0_bm;

	// start conversion
	ADC0.COMMAND |= ADC_STCONV_bm;

	while(!timer_after_pass){
		;// waiting for timer
	}
	cli();
}
```

Εννοποιημένος Κώδικας (1-3 ερωτήματα)

```
void ADC_init(void)
{

	// initialize ADC
	ADC0.CTRLA |= ADC_RESSEL_10BIT_gc;
	ADC0.CTRLA |= ADC_FREERUN_bm;
	ADC0.CTRLA |= ADC_ENABLE_bm;
	ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc;

	ADC0.DBGCTRL |= ADC_DBGRUN_bm;

	// limit is 5
	ADC0.WINLT |= 10;
	ADC0.INTCTRL |= ADC_WCMP_bm;
	ADC0.CTRLE |= ADC_WINCM0_bm;

	sei();
	// START CONVERSION
	ADC0.COMMAND |= ADC_STCONV_bm;
	// wait till find target
	while(!is_robbed){
		;
	}
	cli();

}

void disable_alarm(){
	//Timer set up
	TCA0.SINGLE.CNT = 0; //Clear counter
	TCA0.SINGLE.CTRLB = 0; //Normal Mode
	TCA0.SINGLE.CMP0 = 10; //Stop turning when this value is reached
	TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV1024_gc;
	TCA0.SINGLE.CTRLA |= 1;
	TCA0.SINGLE.INTCTRL = TCA_SINGLE_CMP0_bm;
	cli();

	// start conversion
	ADC0.COMMAND |= ADC_STCONV_bm;
	sei();
	while(!is_robbed_timer){
		// waiting for timer
	}
	cli();
}
```

Εννοποιημένος Κώδικας (1-3 ερωτήματα)

```
void siren(){
	/* Load the Compare or Capture register with the timeout value*/
	TCB0.CCMP = 0x1020;
	TCB0.CNT = 0;
	/* Enable TCB3 and Divide CLK_PER by 2 */
	TCB0.CTRLA |= TCB_ENABLE_bm;
	TCB0.CTRLA |= TCB_CLKSEL_CLKDIV2_gc;
	TCB0.INTCTRL |= TCB_CAPT_bm;

	/* Enable Pin Output and configure TCB in 8-bit PWM mode */
	TCB0.CTRLB |= TCB_CCMPEN_bm;
	TCB0.CTRLB |= TCB_CNTMODE_PWM8_gc;
}

ISR(TCA0_CMP0_vect){
	//Clear timer interrupt flag
	int intflags = TCA0.SINGLE.INTFLAGS;
	TCA0.SINGLE.INTFLAGS = intflags;

	if(pass_input == 1){
		//Disable Timer
		TCA0.SINGLE.CTRLA = 0;
		// turn on flag to flag time to leave home is over
		timer_after_pass = 1;
		// clear correct password flag
		pass_input = 0;
	}
	else if(pass_input == 0){
		siren();
	}
}

ISR(ADC0_WCOMP_vect){
	// clear flags
	int intflags = ADC0.INTFLAGS;
	ADC0.INTFLAGS = intflags;

	// set is robbed variable = 1
	is_robbed = 1;

	is_robbed_timer = 0;

	// turn LED0 on
	_delay_ms(1);
	PORTD.OUTCLR = PIN0_bm;
}
```

Εννοποιημένος Κώδικας (1-3 ερωτήματα)

```
ISR(PORTF_PORT_vect){
	/* Get flags to check for left or right */
	int intflags = PORTF.INTFLAGS;

	/* If switch5 */
	int sw5 = intflags &= ~(00100000);
	int sw6 = intflags &= ~(01000000);

	if(sw6 == 64 && pass_counter == 0 || pass_counter == 3){
		pass_counter++;
	}
	else if(sw5 == 32 && pass_counter == 1 || pass_counter == 2){
		pass_counter++;
	}
	else{
		pass_counter = 0;
		tries++;
	}

	if (tries == 3){
		siren();
	}
	else if(pass_counter == 4){
		// clear pass_counter
		pass_counter = 0;
		is_robbed_timer = 1;
		pass_input = 1;
	}

	// clear flags
	PORTF.INTFLAGS = intflags;

}


ISR(TCB0_INT_vect)
{
	if (pass_counter != 4){
		/* if even, turn on */
		if (pulses % 2 == 0)
			PORTD.OUTCLR = PIN0_bm;
		else
			PORTD.OUT |= PIN0_bm;
		pulses++;
	}
	else{
		// disable pwm
		TCB0.CTRLA = 0;
	}
	TCB0.INTFLAGS = TCB_CAPT_bm; /* Clear the interrupt flag */
	PORTB.IN = PIN5_bm;
}
```

Σχόλια & Παραδοχές Κώδικα

- ## Σημαντικές Μεταβλητές
  - **pass_counter** → Αρχικά είναι «0» και είναι υπεύθυνη για τη σωστή σειρά εισαγωγής των ψηφίων. Οταν γίνει «4» σηματοδοτεί οτι ολοκληρώθηκε η εισαγωγή του κωδικού.
  - ***pass_input*** → Αρχικά είναι «0» και γίνεται «1» όταν εισαχθεί ο κωδικός σωστά.
  - ***timer_pass*** → Αρχικά είναι «0» και γίνεται «1» όταν λήξει το χρονικό περιθώριο για να φύγουν οι ιδιωκτήτες.
  - ***is_robbed*** → Αρχικά είναι μηδέν και μόλις ανιχνευθεί κοντινή κίνηση, δηλαδή τιμή κάτω απο το threshold του ADC WINLT γίνεται «1».
  - ***is_robbed_timer*** → Αρχικά είναι «0» και γίνεται «1» όταν λήξει το χρονικό περιθώριο για να εισαχθεί ο σωστός κωδικός πριν την έναρξη της σειρήνας.
  - ***pulses*** → Χρησιμοποιείται για την ανίχνευση παλμού στον PWM ελέγχοντας τη συμμετρία της τιμής της.

- ## Λετουργία αναμονής εισαγωγής σωστού συνδυασμού και ενεργοποίηση - απενεργοποίηση του χρονιστή

Για τις λειτουργίες αυτές υπεύθυνη είναι η συνάρτηση enable_alarm(). Η συνάρτηση enable_alarm() είναι η πρώτη που καλέιται απο τη main αφότου μόλις ενεργοποιήσει τα switch interrupts για τα SW5, SW6. Κατά την είσοδο ενεργοποιούνται τα ISR Interrupts και «κλειδώνει» εκεί μέχρις ώτου να πατηθεί ένας από τους παραπάνω διακόπτες. Κατα το πάτημα, καλείται η `ISR(PORTF_PORT_vect)` όπου μία απο τις βασικές τις λειτουργίες είναι ο έλεγχος σωστής εισαγωγής κωδικού.

Σχόλια & Παραδοχές Κώδικα

- **Λογική εισαγωγής κωδικού:**

Κάθε φορά που καλείται η `ISR(PORTF_PORT_vect)`, με masking ελέγχεται ποιο switch πατήθηκε. Γνωρίζοντας λοιπόν τη σωστή ακολουθία του κωδικού (sw6-sw5-sw5-sw6) και ότι το πάτημα του sw6 αντιστοιχεί στην $0^{η}$ ή $3^{η}$ κλήση της ISR ενώ το sw5 στή $1^{η}$ ή $2^{η}$, με μία δομή if() else if() εύκολα κάνουμε αυτό τον έλεγχο. Εκτελόντας τη κατάλληλη συνθήκη σε κάθε κάλεσμα της ISR, αυξάνουμε τον pass_counter κατά «1». Κατά την έξοδο από την ISR «καθαρίζουμε» τα PORTF.FLAGS για την εισαγωγή του επόμενου ψηφίου. Όταν ο pass_counter γίνει ίσος με «4», σηματοδοτεί οτι όλα τα ψηφία μπήκαν με τη σωστή σειρά και το pass_input γίνεται «1» ξεκλειδώνοντας έτσι τη συνάρτηση enable_alarm() από την αναμονή εισαγωγής κωδικού. Αν κατά την εισαγωγή ψηφίων η σωστή ακολουθία «χαλάσει», ο pass_counter μηδενίζεται και η λογική αυτή επαναλαμβάνεται.

- **Λογική εν/απ-ενεργοποίσης χρονιστή αποχώρησης**

Μετά τη σωστή εισαγωγή του κωδικού, η enable() αρχικοποιεί τον TCA0 timer και ορίζει μια χρονική ποσότητα για την έξοδο των χειριστών απο τον χώρο. Κατά τη λήξη του, καλείται η `ISR(TCA0_CMP0_vect)` και ελέγχει άν η πυροδότηση της προήλθε από την αναμονή αποχώρησης (if pass_input==1) (κάνουμε αυτό το διαχωρισμό γιατί στη συνέχεια ο TCA0 χρησιμοποιείται και για το περιθώριου χρόνου εισαγωγής του κωδικού πρίν την ενεργοποίηση της σειρήνας). Κατά την εκτέλεση της, καθαρίζουμε και απενεργοποιούμε τον TCA0, επαναφέρουμε την μεταβλητή pass_input=0, δηλώνοντας οτι η επόμενη λειτουργία απαιτεί ξανά την εισαγωγή κωδικού και τέλος κάνουμε την timer_after_pass = 1 δηλώνοντας οτι το χρονικό όριο αποχώρησης τελείωσε. Το ροή πρόγραμματος επιστρέφει στην enable_alarm() και αμέσως μετά στη main().

Σχόλια & Παραδοχές Κώδικα

- **Ενεργοποίηση ADC, LED0, Timer και έλεγχος εισαγωγής συνδυασμού στο χρονικό περιθώριο**

    - **Λογική ADC και LED0**

    Κατά την επιστροφή του προγράμματος στη main, καλείται η συνάρτηση ADC_init(). Εκτελόντας την ADC_init(), αρχικοποιείται ο ADC και θέτει WINLT=10. Στη συνέχεια ενεργοποιεί τα ISR Interrupts, ξεκινάει τη μετατροπή και όσο το WINLT >10 "εξαιτίας" της is_robbed περιμένει σε αυτή τη κατάσταση. Κατά την ανίχνευση και μετατροπή τιμής εισόδου <=10, πυροδοτείται η `ISR(ADC0_WCOMP_vect)` Κατα την εκτέλεση της, καθαρίζει τα flags του ADC, θέτει το is_robbed=1 για να συνεχίσει το επόμενο απαραίτητο βήμα και ανάβει το LED0. Η ροή προγράμματος επιστρέφει στην ADC_init() και ύστερα στη main().

Σχόλια & Παραδοχές Κώδικα

- **Λογική ενεργοποιησης Timer πρίν τη σειρήνα και εισαγωγή 3^{ων} προσπαθειών**

Επιστρέφοντας στη main(), καλείται η συνάρτηση disable_alarm(). Κατά την εκτέλεση της, αρχικοποιείται και ενεργοποιείται ο TCA0 με χρονικό περιθώριο 10 μονάδων για τη σωστή εισαγωγή του κωδικού. Το πρόγραμμα περιμένει σε αυτό το σημείο με τη βοήθεια της μεταβλητής is_robbed_timer μέχρις ώτου να εισαχθεί ο σωστός κωδικός ή να λήξει ο χρόνος και να πυροδοτηθεί η ISR(TCA0_CMP0_vect). Πατώντας οποιοδήποτε απο τα δύο απαραίτητα switches, καλείται η ISR(PORTF_PORT_vect) για την εισαγωγή κωδικού με ίδια λογική κατα την ενεργοποίση του συναγερμού. Στη περίπτωση όπου εισαχθεί σωστά ο κωδικός με λιγότερες απο τρείς προσπάθειες, η μεταβλητή pass_input == 1 και κατά τη πυροδότηση της ISR(TCA0_CMP0_vect) μετά τη λήξη του χρόνου, θα ισχύει η 1^{η} συνθήκη. Στη περίπτωση αυτή καθαρίζει τα flags, απενεργοποιέι τον timer και θέτει τη pass_input=0 ώστε να ζητηθεί ξανά κωδικός στην επομένη ενεργοποίσηση του συναγερμού. Η ροή προγράμματος συνεχίζεται στη main() καλόντας ξανά την enable_alarm() όπου θα επαναληφθούν ακριβώς όλα τα βήματα που περιγράφηκαν μέχρι αυτό το σημείο.

Εναλλακτικά, στη περίπτωση που «σκάσει» ο timer (χωρις κωδικό), καλέιται η ISR(TCA0_CMP0_vect) και αυτή τη φορά ισχύει η δεύτερη συνθήκη. Δηλαδή pass_input == 0 το οποίο σηματοδοτεί πώς δεν έχει εισαχθεί σωστός κωδικός. Στο σημέιο αυτό καλείται η siren() συνάρτηση. Τέλος, κάθε φορά που «χαλάει» η σειρά εισαγωγής των ψηφίων, η μεταβλητή tries αυξάνεται κατά «1» και ο pass_counter μηδενίζεται για την επαναφορά διαδικασίας εισαγωγής κωδικού. Στη περίπτωση όπου tries == 3, καλείται η συνάρτηση siren().

Σχόλια & Παραδοχές Κώδικα

- ## PWM σειρήνα και LED0

Κατά την εκτέλεση της συνάρτησης siren(), με τη βοήθεια του TCB0, τον θέτουμε σε PWM-mode με PER=0x10 και T=0x20 και τον ενεργοποιούμε.

- **Λογική Σειρήνας με LED**

Μετά την ενεργοποίηση της σειρήνας, δηλαδή τη παραπάνω αρχικοποίηση του PWM, σε κάθε ολοκλήρωση παλμού του, καλείται η `ISR(TCB0_INT_vect)`. Κατά την εκτέλεση της, αρχικά ελέγχεται αν έχει εισαχθεί σωστός κωδικός με τη βοήθεια της συνθήκης pass_count != 4. Όπως αναφέρθηκε στο 1ο ερώτημα, pass_counter==4 σηματοδοτεί σωστή εισαγωγή κωδικού. Άρα, όσο ισχύει αυτό (δηλαδή όχι σωστός κωδικός), ελέγχοντας τη συμμετρία της τιμής της μεταβλητής pulses, ανάβει και σβήνει αντίστοιχα το LED0 και αυξάνει το pulses κατά «1» σε κάθε κλήση της ISR.

Στη περίπτωση που εισαχθέι σωστά ο κωδικός με το τρόπο που εξηγήθηκε παραπάνω, απενεργοποιείται ο TCB καθώς πλέον pass_counter==4 (σταματάει δηλαδή η σειρήνα) και μέσω της ISR(PORTF_PORT_vect), όταν εισήχθη σωστά ο κωδικός, το is_robbed_timer ισούται με «1» σηματοδοτώνας ότι απενεργοποιήθηκε ο συναγερμός. Η ροή προγράμματος επιστρέφει στη main() και ξανά την enable_alarm() όπου θα επαναληφθούν ακριβώς όλα τα βήματα που περιγράφηκαν μέχρι αυτό το σημείο.