ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ

ΟΜΑΔΑ Β1 - ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 4

Στοιχεία Φοιτητών:

Αλευράς Ηλίας up1069667@upnet.gr 1069667

Σάββας Στυλιανού up1069661@upnet.gr 1069661

Κώδικας υλοποίησης και των 3ων ερωτημάτων

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#define T1 1024
#define SW5_PIN 5  // Switch 5 pin on PORTF
#define SW6_PIN 6  // Switch 6 pin on PORTF
volatile int sw5;
volatile int sw6;
volatile int correct_code;
volatile int incorrect_try;
volatile int tim0;
volatile int terminate;
volatile int alarm;
int main(void){
	//interrupt
	PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
	PORTF.PIN6CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;

	//PIN is output
	PORTD.DIR |= 0b00000001; //PIN0_bm right

	//TIMER
	/* Load the Compare or Capture register with the timeout value*/
	TCB0.CCMP |= T1;
	/* Enable Capture or Timeout interrupt */
	TCB0.INTCTRL |= TCB_CAPT_bm;
	TCB0.CTRLB |= TCB_CNTMODE_INT_gc;

	//initialize the ADC for Free-Running mode
	ADC0.CTRLA |= ADC_RESSEL_10BIT_gc; //10-bit resolution
	ADC0.CTRLA |= ADC_FREERUN_bm; //Free-Running mode enabled
	ADC0.CTRLA |= ADC_ENABLE_bm; //Enable ADC
	ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc; //The bit //Enable Debug Mode
	ADC0.DBGCTRL |= ADC_DBGRUN_bm; //Window Comparator Mode
	ADC0.WINLT |= 10; //Set threshold
	ADC0.INTCTRL |= ADC_WCMP_bm; //Enable Interrupts for WCM
	ADC0.CTRLE |= ADC_WINCM0_bm; //Interrupt when RESULT < WINL

	//prescaler=1024
	//prescaler=1024
	TCA0.SINGLE.CTRLA=TCA_SINGLE_CLKSEL_DIV1024_gc;
	TCA0.SINGLE.PER = 254; //select the resolution
	TCA0.SINGLE.CMP0 = 127; //select the duty cycle
	//select Single_Slope_PWM
	TCA0.SINGLE.CTRLB |= TCA_SINGLE_WGMODE_SINGLESLOPE_gc;
	//enable interrupt Overflow
	TCA0.SINGLE.INTCTRL = TCA_SINGLE_OVF_bm;
	//enable interrupt COMP0
	TCA0.SINGLE.INTCTRL |= TCA_SINGLE_CMP0_bm;



	terminate=0;
	sei();
	while (terminate==0){

		tim0=0;
```

```
		sw5=0;
		sw6=0;
		correct_code=0;
		incorrect_try=0;
		alarm=0;
		while(correct_code==0 && terminate==0){//anamoni gia isagogi sindiasmou
			;
		}
		correct_code=0;
		incorrect_try=0;
		terminate=0;
		sw5=0;
		sw6=0;
		//energopiisi timer
		TCB0.CTRLA |= TCB_CLKSEL_CLKDIV1_gc ;
		TCB0.CTRLA |= TCB_ENABLE_bm;
		while (tim0==0){//perimene timer
			;
		}
		tim0=0;
		//energopiisi sinagermou
		ADC0.CTRLA |= ADC_ENABLE_bm; //start conversion
		ADC0.COMMAND |= ADC_STCONV_bm; //Start Conversion
		while(alarm==0){//perimene RESULT<WILNT
			;
		}
		//entopismos mikroteris timis apo to katofli
		alarm_on();
		//energopiisi timer
		TCB0.CTRLA |= TCB_CLKSEL_CLKDIV1_gc ;
		TCB0.CTRLA |= TCB_ENABLE_bm;
		//anamoni gia isagogi sindiasmou i 3 lathos prospathies i liksi timer
		while(correct_code==0 && terminate==0 && incorrect_try < 3 && tim0==0){
			;
		}
		if (incorrect_try >= 3 || tim0==1 ){//mpeni mono otan liksi timer i 3
lathos prospathies
			TCA0.SINGLE.CTRLA |= TCA_SINGLE_ENABLE_bm; //Enable
			while(correct_code==0 && terminate==0){//anamoni gia isagogi
sindiasmou
				;
			}
		}
		correct_code=0;
		terminate=0;
		alarm_off();
		TCB0.CTRLA &= ~TCB_ENABLE_bm;
		// To stop the PWM signal
		TCA0.SINGLE.CTRLA &= ~TCA_SINGLE_ENABLE_bm;
	}


}
ISR(PORTF_PORT_vect){
	if ((PORTF.INTFLAGS & (1 << SW5_PIN)) == (1 << SW5_PIN)){
		//clear the interrupt flag
		int z = PORTF.INTFLAGS;
		PORTF.INTFLAGS=z;
		if (sw5==0 && sw6==0){
			sw5=1;
			}else if (sw5==1 && sw6==1){
			sw5=2;
```

```
                        }else{
                        incorrect_try++;//lathos kodikos
                        sw5=0;
                        sw6=0;
                }


                }else if ((PORTF.INTFLAGS & (1 << SW6_PIN)) == (1 << SW6_PIN)){
                //clear the interrupt flag
                int z = PORTF.INTFLAGS;
                PORTF.INTFLAGS=z;
                if (sw5==1 && sw6==0){
                        sw6=1;
                        }else if (sw5==2 && sw6==1){
                        sw6=2;
                        correct_code=1;
                        terminate=1;
                        }else{
                        incorrect_try++;//lathos kodikos
                        sw5=0;
                        sw6=0;
                }

        }

}
ISR(TCB0_INT_vect){
        TCB0.INTFLAGS |= TCB_CAPT_bm; /* Clear the interrupt flag */
        TCB0.CNT = 0b00000000;
        TCB0.CTRLA &= ~TCB_ENABLE_bm;
        tim0=1;

}
ISR(ADC0_WCOMP_vect){
        int intflags = ADC0.INTFLAGS;
        ADC0.INTFLAGS = intflags;
        alarm=1;
        ADC0.CTRLA &= ~ADC_ENABLE_bm; //stop conversion
}
//PWM falling edge
ISR(TCA0_OVF_vect){
        //clear the interrupt flag
        int intflags = TCA0.SINGLE.INTFLAGS;
        TCA0.SINGLE.INTFLAGS = intflags;
        alarm_off();
}

//TIMER (PWM rising edge)
ISR(TCA0_CMP0_vect){
        //clear the interrupt flag
        int intflags = TCA0.SINGLE.INTFLAGS;
        TCA0.SINGLE.INTFLAGS = intflags;
        alarm_on();
}


void  alarm_on(){
        //on
        PORTD.OUTCLR= 0b00000001; //PIN0_bm
}
void alarm_off(){
        //LED is off
```

```
        PORTD.OUT |= 0b00000001; //PIN0_bm
}
```

Αναφορά:

Αρχικά δηλώσαμε ένα TCB0 για timer, τον TCA0 για PWM και τον ADC για αισθητήρα. Αφού το πρόγραμμα μας εισέλθει στην while το πρόγραμμα αρχικοποιει όλες οι μεταβλητές μας. Στην συνέχεια υπάρχει μια while από την οποία το πρόγραμμα μας βγαίνει μόνο αν γίνει ο σωστός συνδυασμός sw5 και sw6. Όταν γίνει interrupt στο INTFLAGS του PORTF το πρόγραμμα μας εισέρχεται στην ISR οπού γίνεται έλεγχος πιο κουμπί πατήθηκε. Ανάλογα με το πιο κουμπί πατήθηκε γίνετε ο έλεγχος των μεταβλητών sw5 και sw6 ώστε να βρεθεί ο κατάλληλος συνδυασμός.

Βήματα if για την εύρεση συνδυασμού :

1. Αν πατηθεί το bit5 γίνεται έλεγχος πιο κουμπί πατήθηκε. Ακόλουθος γίνετε έλεγχος αν τα sw5 και sw6 είναι 0. Αφού είναι 0 sw5 γίνεται ίσο με 1. Αφτί μας δηλώνει ότι είναι το πρώτο κουμπί που έχει πατηθεί και έχει πατηθεί μια φορά.
2. Στην συνέχεια αν πατηθεί το bit6 γίνετε έλεγχος πιο κουμπί πατήθηκε, εντοπίζετε ότι είναι το 6 και το πρόγραμμα εισέρχεται στην if. Αρχικά γινάτια έλεγχος αν sw5 έχει πατηθεί μια φορά, δηλαδή είναι ίσο με 1 και αν το sw6 δεν έχει πατηθεί καθόλου, δηλαδή είναι ίσο με 0. Ακόλουθος το πρόγραμμα μας κάνει την μεταβλητή sw6 ίσή με 1.
3. Αν πατηθεί το κουμπί 5 ξανά το πρόγραμμα μας ελέγχει πιο κουμπί πατήθηκε. Ακόλουθος ελέγχει αν το sw5=1 και sw6=1. Αυτό σημαίνει ότι έχουν πατηθεί από μια φορά. Το πρόγραμμα μας αλλάζει την τιμή του sw5 σε 2 που δηλώνει ότι έχει πατηθεί 2 φορές.
4. Ακόλουθος αν πατηθεί το κουμπί 6 γίνετε έλεγχος πιο κουμπί πατήθηκε. Μετα γίνεται έλεγχος αν το sw5=2 και sw6=1. Αυτό δηλώνει ότι ο συνδυασμός είναι σωστός και οι τιμές correct_code και terminate γίνονται ίσες με 1 ώστε να βγει το πρόγραμμα μας από την while.
5. Σε όπιο από τα βήματα 1, 2, 3, 4 γινεί λάθος συνδυασμός το πρόγραμμα εισέρχεται στην else οπού οι μεταβλητές sw5, sw6 γίνονται ισες με 0 και προσθέτουμε 1 στην incorrect_try.

Αφού γίνει σωστός συνδυασμός ενεργοποιείται ο TCB0 και το πρόγραμμα μας εισέρχεται μέσα σε μια while οπού τον περιμένει να τελειώσει. Αφού τελειώσει το πρόγραμμα μας ενοποιεί τον ADC και εισέρχεται σε μια while οπού περιμένει να γίνει interrupt από αυτόν. Αφού γίνει interrupt το LED0 ανάβει μέσο της συνάρτησης alarm_on() και ενεργοποιείται ο TCB0. Στην συνέχει το πρόγραμμα μας εισέρχεται μέσα σε μια while οπού για να βγει από αυτήν πρέπει, είτε να γίνει ο σωστός συνδυασμός, είτε να γίνουν 3 λάθος προσπάθειες, είτε να λήξει ο timer. Αφού βγει από την while γίνεται έλεγχος με μια if για το αν έχουν γίνει 3 λάθος προσπαθείς η αν έχει λήξει ο timer. Αν έχει γίνει κάποιο από αυτά εισέρχεται στην if οπού ενεργοποιείται ο PWM και το πρόγραμμα μας μπαίνει σε μια while στην οποία για να βγει πρέπει να γίνει ο κατάλληλος συνδυασμός.