Εθνικό **Μ**ετσόβιο **Π**ολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εργαστήριο Μικροϋπολογιστών, ακ. έτος 2024 - 2025

Λύσεις στην 3η Εργαστηριακή Άσκηση

Όνομα: Ειρήνη	Όνομα: Γεώργιος
Επώνυμο: Σιμιτζή	Επώνυμο: Οικονόμου
ΑΜ: 03121063	ΑΜ: 03121103
Εργαστηριακή Ομάδα: 30	

Ζήτημα 3.1

Στο ζήτημα αυτό υλοποιούμε κώδικα Assembly ο οποίος μεταβάλλει την φωτεινότητα ενός LED αναλόγως το Duty Cycle της PWM κυματομορφής. Τα επιτρεπτά επίπεδα των τάξεων 2%, 10%, …, 98% αποθηκεύονται στον πίνακα matrix στην Program Memory. Υπολογίζονται με μέγιστη τιμή το 255 εφόσον ο καταχωρητής OCR1A του TMR1A είναι 8-bit, δηλαδή, έχει τιμές από 0 μέχρι 255. Παρακάτω παρουσιάζονται τα καίρια κομμάτια του κώδικα.

```
45      main:
46          sbis PIND, 3            ; Check if PD3 is 1
47          rjmp increase           ; Then, if PD3 = 1, increase by 8%
48
49          sbis PIND, 4
50          rjmp decrease
51
52          rjmp main
53
54      load_new_DC_PD3:
55          cpi DC_COUNTER, 12      ; Check if DC_VALUE = 98%
56          breq out_to_leds
57          inc DC_COUNTER
58          ldi ZL, low(2*matrix)   ;  Load low part of byte address into ZL
59          ldi ZH, high(2*matrix)  ; Load high part of byte address into ZH
60
61          add ZL, DC_COUNTER
62
63          lpm DC_VALUE, Z
64          rjmp out_to_leds
65
66      load_new_DC_PD4:
67          cpi DC_COUNTER, 0       ; Check if DC_VALUE = 02%
68          breq out_to_leds
69          dec DC_COUNTER
70          ldi ZL, low(2*matrix)   ;  Load low part of byte address into ZL
71          ldi ZH, high(2*matrix)  ; Load high part of byte address into ZH
72
73          add ZL, DC_COUNTER
74
75          lpm DC_VALUE, Z
76          rjmp out_to_leds
```

Ζήτημα 3.2

Στο ζήτημα αυτό υλοποιούμε κώδικα C παρόμοιο με τον προηγούμενο, όπου υπολογίζουμε εντός του βρόχου την μέση τιμή του ADC χρησιμοποιώντας bitwise πράξεις. Παρακάτω παρουσιάζονται τα καίρια κομμάτια του κώδικα.

```
49      while(1){
50
51          if(reset == 16){
52
53              reset = 0;
54              ADC_value = (ADC_value >> 4); // Divide by 2^4 = 16
55
56              if (ADC_value >= 0 & ADC_value <= 200) {
57                  PORTD = 0x01;
58              }
59
60              else if (ADC_value > 200 & ADC_value <= 400) {
61                  PORTD = 0x02;
62              }
63
64              else if (ADC_value > 400 & ADC_value <= 600) {
65                  PORTD = 0x04;
66              }
67
68              else if (ADC_value > 600 & ADC_value <= 800) {
69                  PORTD = 0x08;
70              }
71
72              else {
73                  PORTD = 0x10;
74              }
75              ADC_value =0;
76          }
77
            delay_ms(100);                  // Call delay
79          ADC_value += adc_read();        // Read ADC_value
```

Ζήτημα 3.3

Στο ζήτημα αυτό υλοποιούμε κώδικα C με δύο τρόπους το mode1, όταν έχει πατηθεί το PIN6 της PORTB και το mode2, όταν έχει πατηθεί το PIN7 της PORTB, αντίστοιχα. Παρόμοια, τα επιτρεπτά επίπεδα των τάξεων είναι αποθηκευμένα στον πίνακα table[]. Κάναμε τον ADMUX left-adjusted και κρατήσαμε τα οκτώ σημαντικότερα ψηφία για τη μετατροπή (αντί για τα δέκα, ώστε να διευκολυνθούμε με τον 8-bit καταχωρητή OCR1AL). Τα δύο LSBs θεωρούμε δεν επηρεάζουν το αποτέλεσμα. Παρακάτω παρουσιάζονται τα καίρια κομμάτια του κώδικα.

```
43      unsigned char MODE = 1;
44      while(1){
45           if (PIND == 0b10111111) {     //PD6
46              MODE = 1;
47          }
48          if (PIND == 0b01111111) {      //PD7
49              MODE = 2;
50          }
51           while(MODE==1){
52          if((PIND & 0x02) == 0x00) {    // PD1 is pressed
53             do {                        // wait while pressed
54                 _delay_ms(5);
55               } while((PIND & 0x02) == 0x00);
56
57           if(index == 0) continue;      // don't increase at 98%
58              OCR1AH = 0x00;             // values from 0 to 255
59              OCR1AL = table[--index];
60          }
61
62           if((PIND & 0x04) == 0x00) {        // check if PD2 pressed
63              do {                            // (logical 0)
64                  _delay_ms(5);
65              } while((PIND & 0x04) == 0x00); // while being pressed wait
66              if(index == 12) continue;       // don't decrease at  2%
67              OCR1AH = 0x00;
68              OCR1AL = table[++ index];
69          }
70          if (PIND == 0b01111111) {           //PD7
71                  MODE = 2;
72          }}
73          while (MODE=2){
74             OCR1AL = adc_read() >> 2;
75              }
76          }
77      }
```