4η Εργαστηριακή Αναφορά

Παπαδόπουλος Χαράλαμπος 03120199

Στρίφτης Γεώργιος 03121200

**Άσκηση 1**

Σκοπός της άσκησης ήταν η εξοικείωση μας με την οθόνη της πλακέτας NTUA\_BOARD μέσω ενός προγράμματος υπολογισμού τάσης στον ADC.

Παρακάτω παρατίθεται ο assembly κώδικας (παραλείποντας τις αρχικοποιήσεις της οθόνης καθώς και την υλοποίηση της καθυστέρησης καθώς είναι όπως στις σημειώσεις):

|  |
| --- |
| main:  ldi voltage, 5  rcall lcd\_clear\_display  lds r17, ADCSRA  ori r17,(1<<ADSC)  sts ADCSRA, r17 ; start conversion  MUL ADC\_L, voltage ;multiply 8 lsb with voltage  MOV R30, R0  MOV R31, R1    ;we want to add 8 zeros(multiply by 2^8) to the high bit after we multiply it by the voltage  MUL ADC\_H, voltage  add r31, r0    ; r31-r30 hold the wanted value    mov r29, r31 ; We take the first 6 bits(integer part)  lsr r29  lsr r29 ;int part saved in r29      ;multiply lower 10 bits by 100 and divide by 1024 to get 2 fractional digits precision  ;first multiply low bit  ldi r16, 100  mul r30, r16  mov r28, r1 ;keep high bit of multiplication(it's like we divided by 2^8)  andi r31, 0b00000011 ;keep last 2 bits(2 MSBS of fractional parts)  mul r31, r16  clc  ror r1  ror r0  ror r1  ror r0  clc  lsr r28  lsr r28  clc  add r28, r0 ;keep low byte of multiplication    ;fraction part saved in r28    fraction\_convert:  ;convert to bcd and load integer part to display  mov r24, r29  rcall bcd\_conversion  mov r24, r26  call load\_to\_display  ;load '.' to display  ldi r24, '.'  call lcd\_data  ;load fraction part to r24  mov r24, r28  rcall bcd\_conversion  ;load tens bcd to display  mov r24, r17  rcall load\_to\_display  ;load units bcd to display  mov r24, r26  rcall load\_to\_display  ldi r24, low(1000)  ldi r25, high(1000)  rcall wait\_msec  rjmp main          ADC\_INT:  lds ADC\_L, ADCL  lds ADC\_H, ADCH  out PORTD, ADC\_H  reti      bcd\_conversion:  clr R17 ; Clear R17 for tens  clr r26 ; Clear R18 for units  loop:  cpi r24, 10  brlo done ; If less than 10, branch to done  ; Subtract 10 from R16 and increment R17 (tens digit)  subi R24, 10  inc R17  rjmp loop ; Repeat until R16 is less than 10  done:  ; Now R16 contains the units (0-9), so copy it to R18  mov r26, R24  ret |

Το βασικότερο πρόβλημα που αντιμετωπίσαμε ήταν στην μετατροπή του δυαδικού αριθμού σε δεκαδική μορφή καθώς είχαμε έναν 10-bit αριθμό.

Η λύση μας ήταν να υλοποιήσουμε μια απλή υπορουτίνα, μέσω της οποίας βρίσκουμε τις δεκάδες και τις μονάδες του κλασματικού μέρους του αριθμού, τις οποίες ύστερα μετατρέπουμε σε δεκαδική μορφή για ανάγνωση από την οθόνη μέσω μιας σειράς από if-else.

|  |
| --- |
| load\_to\_display:  ;take input from r24 and load to display  cpi r24, 0  brne one  ldi r24, '0'  call lcd\_data  rjmp end  one:  cpi r24, 1  brne two  ldi r24, '1'  call lcd\_data  rjmp end  two:  cpi r24, 2  brne three  ldi r24, '2'  call lcd\_data  rjmp end  three:  cpi r24, 3  brne four  ldi r24, '3'  call lcd\_data  rjmp end  four:  cpi r24, 4  brne five  ldi r24, '4'  call lcd\_data  rjmp end  five:  cpi r24, 5  brne six  ldi r24, '5'  call lcd\_data  rjmp end  six:  cpi r24, 6  brne seven  ldi r24, '6'  call lcd\_data  rjmp end  seven:  cpi r24, 7  brne eight  ldi r24, '7'  call lcd\_data  rjmp end  eight:  cpi r24, 8  brne nine  ldi r24, '8'  call lcd\_data  rjmp end  nine:  ldi r24, '9'  call lcd\_data    end:  ret |

**Άσκηση 2**

Το ζητούμενο της δεύτερης άσκησης ήταν να υλοποιήσουμε τον ίδιο κώδικα σε c. Εδώ η βασικότερη δυσκολία ήταν η «μετάφραση» της αρχικοποίησης της οθόνης, από assembly σε c.

|  |
| --- |
| uint16\_t read\_adc() {  ADMUX |= (1 << MUX0);  ADCSRA |= (1 << ADSC);  while (ADCSRA & (1 << ADSC));  return ADC;  }  int main() {  setup();  while(1) {  lcd\_clear\_display();  double adc = read\_adc();  double Vin = (adc \* VOLTAGE) / 1024;  char result[5];  sprintf(result, "%.2f", Vin);  for(int i = 0; i < 4; i++) {  lcd\_data(result[i]);  }  \_delay\_ms(100);  }  }  void setup() {  DDRD = 0xFF;  lcd\_init();  \_delay\_ms(100);  lcd\_clear\_display();  ADMUX = 0b01000001;  ADCSRA = 0b10000111;  }  void lcd\_init() {  \_delay\_ms(200);  write\_2\_nibbles(0x30);  \_delay\_ms(5);  write\_2\_nibbles(0x30);  \_delay\_us(200);  write\_2\_nibbles(0x30);  \_delay\_us(200);  write\_2\_nibbles(0x20); // Switch to 4-bit mode  \_delay\_us(200);  lcd\_command(0x28);  lcd\_command(0x0C);  lcd\_clear\_display();  lcd\_command(0x06);  }  void lcd\_clear\_display() {  lcd\_command(0x01);  \_delay\_ms(2);  }  void lcd\_command(uint8\_t cmd) {  PORTD &= ~(1 << PD2);  write\_2\_nibbles(cmd);  \_delay\_us(50);  }  void lcd\_data(char data) {  PORTD |= (1 << PD2);  write\_2\_nibbles(data);  \_delay\_us(50);  }  void write\_2\_nibbles(uint8\_t data) {  uint8\_t low\_bits = PORTD & 0x0F;  // Send high nibble  PORTD = (data & 0xF0) | low\_bits;  PORTD |= (1 << PD3);  \_delay\_us(1);  PORTD &= ~(1 << PD3);  \_delay\_us(50);  // Send low nibble  PORTD = ((data << 4) & 0xF0) | low\_bits;  PORTD |= (1 << PD3);  \_delay\_us(1);  PORTD &= ~(1 << PD3);  } |

**Άσκηση 3**

Η άσκηση αυτή προσομοιώνει την λειτουργία ενός αισθητήρα μονοξειδίου του άνθρακα μέσω του επιπέδου τάσης του ADC3 που ρυθμίζεται με ποτενσιόμετρο. Καθώς η έξοδος μας είναι στα λαμπάκια επιλέξαμε ισομοιρασμό των επιπέδων και βήμα 83ppm για το πότε θα ανάψει το επόμενο.

Παρατίθεται το βασικό κομμάτι της άσκησης καθώς είναι κατά κύριο λόγο βασισμένη στον κώδικα της προηγούμενης.

|  |
| --- |
| int main() {  int danger = 0;  setup();  while(1) {  lcd\_clear\_display();  double adc = read\_adc();  double ppm = (adc \* 500) / 1024;  if(ppm >= 70) danger = 1;  else danger = 0;  if(danger == 0){  PORTB = 1;  char message[5] = "CLEAR";  for (int i = 0; i < 5; i++)  lcd\_data(message[i]);  \_delay\_ms(100);  }  else {  PORTB = 0;  char message[12] = "GAS DETECTED";  for(int i = 0; i < 12; i++){  lcd\_data(message[i]);  }  \_delay\_ms(50);  if(ppm <= 83) PORTB = 1;  else if(ppm <= 166) PORTB = 3;  else if(ppm <= 249) PORTB = 7;  else if(ppm <= 332) PORTB = 15;  else if(ppm <= 415) PORTB = 31;  else PORTB = 63;  \_delay\_ms(50);  }  }  } |