2023

Αναφορά 3ης εργαστηριακής Άσκησης

Εργαστήριο μικροϋπολογιστών

ΠΑΠΑΔΟΠΟΥΛΟΣ ΣΠΥΡΙΔΩΝ (ΑΜ):03120033
ΕΜΜΑΝΟΥΗΛ ΞΕΝΟΣ (ΑΜ):03120850

Ζήτημα πρώτο

Στο ερώτημα αυτό μας ζητήθηκε να υλοποιήσουμε σε assembly μία ένα πρόγραμμα το οποίο θα χρησιμοποιεί την δυνατότητα του επεξεργαστή μας ATmega328PB να παράγει PWM σήματα. Αυτό γίνεται χρησιμοποιώντας ένα εσωτερικό timer του επεξεργαστή(TMR1A) το οποίου την λειτουργία εμείς καθορίζουμε με βάση κάποια flags που θα θέσουμε(αναλύονται παρακάτω) σε κάποιους εσωτερικούς καταχωρητές του επεξεργαστή. Αναλυτικότερα η λειτουργία για την παραγωγή του PWM σήματος γίνεται ως εξής: Ο timer ξεκινάει από μία τιμή την οποία εμείς καθορίζουμε μέσω των flags που θα θέσουμε και μετράει προς τα πάνω ή προς τα κάτω. Ο timer για κάθε τιμή που παίρνει συγκρίνεται με μία τιμή η οποία περιέχεται σε έναν διπλό καταχωρητή ,τον OCR1A. Την τιμή του OCR1A την θέτουμε εμείς. Από την σύγκριση της τιμής του timer με τον OCR1A καταλήγουμε στο αν η έξοδος του PWM σήματος θα είναι 1 ή 0 και συνεπώς έτσι καθορίζεται η βασική παράμετρος ενός PWM σήματος που είναι το duty cycle. Εμείς μπορούμε να καθορίσουμε με βάση κάποια flags εάν το αποτέλεσμα της σύγκρισης θα οδηγεί σε έξοδο ίση με το 0 ή το 1(δηλαδή εάν όταν ο timer θα είναι μικρότερος από τον OCR1A εάν το αποτέλεσμα θα είναι 0 ή 1).Με βάση αυτά μπορούμε να παράγουμε χρησιμοποιώντας την πλακέτα μας ένα PWM σήμα το οποίο θα δώσουμε σαν έξοδο στο led PB1. Προφανώς όσο μεγαλύτερο το duty cycle τόσο μεγαλύτερη θα είναι και η φωτεινότητα του LED. Αυτό είναι ουσιαστικά και αυτό που μας ζητήθηκε στο πρώτο ζήτημα, να χρησιμοποιήσουμε την γεννήτρια κυματομορφών του επεξεργαστή μας με σκοπό να ελέγξουμε την φωτεινότητα του PB1.

Στο πλαίσιο της υλοποίησης δεν υπάρχει κάποια ιδιαιτερότητα στον κώδικα. Αρχικά, θα εξηγήσουμε ποια flags θέσαμε για να πραγματοποιήσουμε την επιθυμητή λειτουργία. Θα πρέπει να θέσουμε τα flags WGM10 και WGM12 για να θέσουμε το mode λειτουργίας σε γρήγορο PWM , 8 bit. Επίσης θέτουμε το COM1A1 για να καθορίσουμε μη αναστρέφουσα λειτουργία(δηλαδή το OC1A/B έχει μία ελάχιστη τιμή BOTTOM στην οποία το duty cycle του PWM σήματος είναι το ελάχιστο και αντίθετα όταν έχει την τιμή TOP το duty cycle είναι το μέγιστο. Η τιμή bottom προφανώς για εμάς είναι 0 και η τιμή TOP του OCR1A καθορίζεται από το mode λειτουργίας . Εδώ έχουμε 8-bit mode οπότε η μέγιστη τιμή είναι η 2^8-1=255). Τέλος, θέτουμε και το CS11 προκειμένου να καθορίσουμε την συχνότητα αύξησης του TCNT1 (εδώ θα έχουμε 16/8=2Mhz). Στην συνέχεια παραθέτουμε τον κώδικα και οι υπόλοιπες διευκρινήσεις πάνω σε αυτόν θα παρατεθούν ακολούθως:

```
.include "m328PBdef.inc"

    .def temp=r17
    .def dc_value=r18
    .def counter=r19

    .org 0x00
    jmp reset

Table:
    .dw 5, 26, 46, 66, 87
    .dw 107, 128, 148, 168, 189
    .dw 209, 230, 250
reset:
    ldi temp,high(RAMEND)   ; initialize stack
    out SPH,temp
```

```
    ldi temp,low(RAMEND)
    out SPL,temp
    ldi zh,high(Table*2)
    ldi zl,low(Table*2)
    clr temp
    out DDRD,temp
    adiw zl,12        ; initialize dc value
    lpm
    mov dc_value,r0
    ldi counter,6    ; initialize dc_value level counter
    ldi temp,0b00111111
    out DDRB,temp;    ; set PB5-PB0 as output
    ldi temp, (1<<WGM10) | (1<<COM1A1)  ; set WGM10 to 1 and WGM12 to 1 for
    sts TCCR1A,temp            ; FAST PWM,8-bit mode of operation
    ldi temp, (1<<WGM12) | (1<<CS11)    ;set COM1A1 to 1 for non inverting
mode
    sts TCCR1B,temp            ; set CS11 to one to set the prescalarar to 8
                    ; frequency of timer TCNT1 16MHz/8=2MHz
main_loop:
    sbic PIND,1        ; if PD1 is 1 no increase should occure
    rjmp no_increase
wait_release_PD1:
    sbis PIND,1        ; check if PD1 is still pressed
    rjmp wait_release_PD1
    cpi counter,12
    breq ouput_PWM  ; if we reached the max limit don't increase the counter
    inc counter
    adiw zl,2        ; also increase the array index
    rjmp ouput_PWM
no_increase:
    sbic PIND,2        ; if PD2 is 1 no increase should occure
    rjmp ouput_PWM
wait_release_PD2:
    sbis PIND,2        ; check if PD2 is still pressed
    rjmp wait_release_PD2
    cpi counter,0
    breq ouput_PWM  ; if we reached the min limit don't decrease the counter
    dec counter
    sbiw zl,2        ; also decrease the array index
ouput_PWM:
    lpm
    mov dc_value,r0    ; find the current dc value
    sts ocr1al,dc_value ; and set OCR1AL
    rjmp main_loop
```

Ο κώδικας παρέχει επαρκή σχόλια για να είναι κατανοητός, εντούτοις θα προσθέσουμε εμείς κάποια ακόμα. Αρχικά, στην υλοποίηση μας γράφουμε τον ζητούμενο πίνακα στην μνήμη προγράμματος(που είναι 16-bit) από την οποία αντλούμε τα δεδομένα μέσω της τιμής του καταχωρητή Zl και Ro. Αρχικά θέτουμε τους Zl και Zh στην τιμή TABLE*2 διότι η μνήμη προγράμματος είναι 16 bit. .Στην συνέχεια αφού θέσουμε τις κατάλληλες εισόδους και εξόδους, αρχικοποιήσουμε τα flags και κάποιους καταχωρητές όπως ο counter ξεκινάμε το κύριο πρόγραμμα(ο counter είναι ένας βοηθητικός καταχωρητής τον οποίο τον χρησιμοποιούμε για να βλέπουμε εάν έχουμε φτάσει στα όρια του πίνακα που έχουμε αποθηκεύσει στην μνήμη). Αρχικά ελέγχουμε εάν το PD1 έχει πατηθεί(ελέγχοντας εάν το PB1 είναι 0). Αν έχει πατηθεί αναμένουμε ο χρήστης να το αφήσει και στην συνέχεια ελέγχουμε εάν ο counter μπορεί να αυξηθεί. Εάν μπορεί(δηλαδή δεν έχει φτάσει την μέγιστη τιμή του) αυξάνουμε την τιμή αυτού κατά 1 και του ζεύγους καταχωρητών Zl-Zh κατά 2(αφού δείχνουν σε 16-bit διευθύνσεις στην μνήμη) και τέλος πάμε στην παρουσίαση του αποτελέσματος που είναι στην τρέχουσα θέση μνήμης. Προφανώς το δυικός ανάλογο ισχύει αν πατείται το PD2. Στην φάση του output απλώς παίρνουμε την τρέχουσα τιμή από την program memory και την βάζουμε στον OCR1AL προκειμένου να θέσουμε το κατάλληλο duty cycle.

Ζητούμενο Δεύτερο

Στο ζητούμενο αυτό καλούμαστε να γράψουμε το πρόγραμμα του προηγούμενου ζητήματος σε C που παράγει PWM έξοδο στον ακροδέκτη PB1, αλλά αυτή την φορά ο ADC θα διαβάζει την έξοδο αυτή και ανάλογα με την τάση που διάβασε ο ADC πρέπει να ανάβει και το αντίστοιχο led του PORTD σύμφωνα με τον παρακάτω πίνακα:

Τάση εισόδου στον ADC(Vadc)	LED ON
0 Volt ≤ Vadc ≤ 0,625 Volt	PD0
0,625 Volt < Vadc ≤ 1,25 Volt	PD1
1,25 Volt < Vadc ≤ 1,875 Volt	PD2
1,875 Volt < Vadc ≤ 2,5 Volt	PD3
2,5 Volt < Vadc ≤ 3,125 Volt	PD4
3,125 Volt < Vadc ≤ 3,75 Volt	PD5
3,75 Volt < Vadc ≤ 4,375 Volt	PD6
4,375 Volt < Vadc ≤ 5 Volt	PD7

Ο κώδικας που υλοποιεί τα παραπάνω καθώς και η εξήγησή του ακολουθούν.

```
#define F_CPU 16000000UL
#include "avr/io.h"
#include <util/delay.h>

const uint16_t table[] = {5, 26, 46, 66, 87, 107, 128, 148, 168, 189, 209,
230, 250};


int main()
{
    uint16_t duty;
```

```
    uint16_t temp;

    //set TMRIA in fast PWM 8 bit mode with non-inverted output
    //prescale-8
    TCCR1A = (1<<WGM10) | (1<<COM1A1);
    TCCR1B = (1<<WGM12) | (1<<CS11);

    DDRD |= 0xFF;
    DDRB |= 0b00000010;

    ADMUX |= 0b01000001; //Right adjusted, ADC1
    ADCSRA |= 0b10000111;



    duty=6;
    OCR1AL=table[duty];
    temp=0;
    //PORTD=3;
    //set PB5-PB0 pins as output
    while (1)
    {

        //DDRB |= 0b00111111;
        if((PINB & 16) != 16){
            if(duty<12) OCR1AL=table[++duty];
            while((PINB & 16) != 16) {}
        }else if((PINB & 32) != 32){
            if(duty>0) OCR1AL=table[--duty];
            while((PINB & 32) != 32) {}
        }

        //DDRB |= 0x00;
        _delay_ms(70);
        ADCSRA|= (1<<ADSC);    //Start ADC
        while((ADCSRA & 0x40)==0x40){}   //Wait until ADC is finished

        temp=ADC;
        //PORTD=temp;
        //PORTD=temp;
        if (temp <= 128) PORTD=1;
        else if (temp <= 256) PORTD=2;
        else if (temp <= 384) PORTD=4;
        else if (temp <= 512) PORTD=8;
        else if (temp <= 640) PORTD=16;
        else if (temp <= 768) PORTD=32;
        else if (temp <= 896) PORTD=64;
        else PORTD=128;
```

```
        //PORTD=0xFF;
        //PORTD=temp;
        _delay_ms(100);

    }
}
```

Αρχικά δημιουργούμε τον πίνακα των duty cycle με ίδιες προδιαγραφές σύμφωνα με το προηγούμενο ζήτημα. Έπειτα θέτουμε τις σημαίες για τους καταχωρητές TCCR1A και TCCR1B ώστε να παράγουμε ένα fast PWM 8bit σήμα, θέτουμε κατάλληλα τους ακροδέκτες εξόδους OC1A και OC1B και θέτουμε την συχνότητα λειτουργίας του Timer/Counter1 (TCNT1) ίση με 1/8 του ρολογιού. Στην συνέχεια θέτουμε ως έξοδο την PORTD, το PB1 ως έξοδο και τα υπόλοιπα PINs του PORTB ως είσοδο ώστε να μπορούμε να αυξομειώνουμε το duty cycle του παραγόμενου PWM σήματος. Με το PB4 θα αυξάνουμε το duty cycle του PWM σήματος ενώ με το PB5 θα το μειώνουμε. Επίσης θέτουμε το περιεχόμενο του ADC ως right adjusted, ως κανάλι εισόδου το ADC1 ενώ στον ADCSRA κάνουμε enable το ADC και θέτουμε τον prescaler ίσο με 128. Προτού μπούμε στην κύρια ροή του προγράμματος θέτουμε το duty cycle στην μέση. Αυτό γίνεται μέσω της βοηθητικής μεταβλητής duty που μας βοηθάει ώστε να παίρνουμε την κατάλληλη τιμή από τον πίνακα με τα duty cycles. Έτσι μπαίνουμε σε ένα while loop που εκτελείται συνεχώς. Αρχικά ελέγχεται αν έχει πατηθεί το PB4 . Αν έχει πατηθεί και το duty cycle δεν είναι στο μέγιστο της τιμής του (δηλαδή το duty δεν θα βγει εκτός ορίων του πίνακα table αν αυξηθεί κατά 1) τότε αυξάνουμε το duty κατά 1 και βάζουμε την νέα τιμή στον OCR1AL αλλάζοντας έτσι το duty cycle του PWM σήματος. Στην συνέχεια για να αποφύγουμε 1 πάτημα του PB4 να θεωρηθεί από το πρόγραμμα ως παραπάνω ελέγχουμε αν μετά την αύξηση του duty το PB4 είναι ακόμα πατημένο. Για όσο είναι πατημένο περιμένουμε. Αυτό γίνεται μέσω της εντολής while((PINB & 16) != 16) {}. Παρόμοια με το παραπάνω λειτουργούμε για την μείωση του duty cycle όταν πατιέται το κουμπί PB5. Στην συνέχεια αφού το OCR1AL έχει πάρει την επιθυμητή τιμή ξεκινάμε το ADC και περιμένουμε μέχρι αυτό να τελειώσει. Έπειτα παίρνουμε το αποτέλεσμα του ADC το τοποθετούμε στην 16bit μεταβλητή temp. Σύμφωνα με τον τύπο:

$$ADC = \frac{Vin}{Vref} \cdot 2^n$$

Όπου Vref=5V και n=10 βρίσκουμε την έξοδο του ADC για τα όρια των κατηγοριών. Έτσι τα όρια για τις κατηγορίες είναι 0-128, 129-256, 257-384 κτλ. Έτσι στο πρόγραμμα ανάλογα με την τιμή του ADC έχουμε και την ανάλογη έξοδο μέσω συνεχόμενων ifs. Τα delays μεταξύ του ADC και της εξόδου χρησιμεύουν ώστε η δειγματοληψία που πραγματοποιεί ο ADC να συγχρονίζεται με την έξοδο της γεννήτριας PWM .Αυτό έχει ως αποτέλεσμα τα δείγματα σε κάθε δειγματοληψία να λαμβάνονται με τον ίδιο τρόπο και όχι σε τυχαία σημεία του PWM, γεγονός που μπορεί να οδηγήσει σε διαφορετικά αποτελέσματα του ADC για είσοδο το ίδιο PWM σήμα.

Ζήτημα Τρίτο

Σε αυτό το ζήτημα καλούμαστε να γράψουμε το πρόγραμμα του ζητήματος 3.1 και εμπλουτίζοντάς το. Στο πρόγραμμα πρέπει να υπάρχουν δύο modes (το mode1 και το mode2) τα οποία διαλέγουμε

πατώντας το PD6 και το PD7 αντίστοιχα. Στο mode1 η φωτεινότητα του led αυξάνεται πατώντας το PD1 και μειώνεται πατώντας το PD2. Στο mode2 η φωτεινότητα ελέγχεται από το ποτενσιόμετρο POT1. Ο κώδικας είναι ο ακόλουθος:

```
#define F_CPU 16000000UL
#include "avr/io.h"
#include <util/delay.h>

int main(){
    const uint16_t array[]={5,26,46,67,87,108,128,148,169,189,210,230,251};
    unsigned int counter;
    unsigned int dc_value;
    unsigned int i;
    counter=6;
    DDRD=0;
    TCCR1A = (1<<WGM10) | (1<<COM1A1); // set WGM10 to 1 and WGM12 to 1 for
                        //FAST PWM,8-bit mode of operation
    TCCR1B = (1<<WGM12) | (1<<CS11); //set COM1A1 to 1 for non inverting mode
                          // set CS11 to one to set the prescalar to 8
    DDRB=0b0000010;          //set PORTB as input
    ADCSRA=0b10000111; //set ADEN=1 -> ADC Enable.
                      // ADCS=0 -> No Conversion
                      // set ADIE=0 -> disable adc interrupt
                      // set ADPS=111 ->set the prescalar to 128 =>
fADC=125KHz
    ADMUX=0b01100000;  //set REFSn=01 -> Vref=5V
            // set MUXn=0000 select ADC0(A0)
            // set ADLAR=1-> Left adjust the ADC result
    while(1){
        if((PIND & 64)!=64){
            while((PIND & 128)==128){
                if((PIND & 2)!=2){ // check if PD1 is pressed
                    while((PIND & 2)==2) // wait until PD1 is unpressed
                        ;
                    if(counter!=12)
                        counter++;
                }
                else if((PIND & 4)!=4){ // check if PD2 is pressed
                    while((PIND & 4)==4) // wait until PD2 is unpressed
                        ;
                    if(counter!=0)
                        counter--;
                }
                dc_value=array[counter];
                OCR1AL=dc_value;
            }
        }
        else if((PIND & 128)!=128){
            while((PIND & 64)==64){
```

```
                _delay_ms(100);
                ADCSRA |=(1<<ADSC); // set ADCS flag of ADCSRA flag indicates
that conversion starts
                while(( ADCSRA & (1<<ADSC)) != 0);  // wait until ADSC flags
becomes 0=> conversion was completed
                OCR1AL=ADCH;
            }
        }
    }
}
```

Στην αρχή θέτουμε τον πίνακα με τα DC_Values που θα μας χρειαστεί στο mode1, τα flags των καταχωρητών όπως έχουν ήδη εξηγηθεί (αυτή όμως την φορά θέτουμε ως είσοδο του ADC το ADC0 και το αποτέλεσμα του ADC ως left adjusted), την θύρα PORTD ως είσοδο και το pin PB1 ως έξοδο. Στην συνέχεια μπαίνουμε σε ένα while loop που εκτελείται συνεχώς. Έπειτα μέσω του if που συναντάμε διαλέγουμε mode. Αν επιλέξουμε το mode1 μπαίνουμε στο πρώτο if το οποίο εκτελείται συνεχώς μέχρι να πατηθεί το PD7. Καθώς εκτελείται το πρόγραμμα στο mode1 το σκεπτικό λειτουργίας του είναι παρόμοιο με αυτό που ακολουθήθηκε για την αυξομείωση του duty cycle στο ζήτημα 2. Αν πατηθεί το PD7 τότε «μπαίνουμε» στο else και εκτελείται το mode2 μέχρι να πατηθεί το PD6 που θα εξέλθουμε από το mode2 και θα μπούμε στο mode1. Όσο είμαστε στο mode2 διαβάζουμε την τιμή του ποτενσιομέτρου που είναι συνδεδεμένο στο ADC0 και στην συνέχεια παίρνουμε την τιμή του ADCH δηλαδή τα 8 MSB του αποτελέσματος του ADC (το αποτέλεσμα είναι Left Adjusted). Το αποτέλεσμα του «περνάμε» κατευθείαν στο OCR1AL αλλάζοντας έτσι την φωτεινότητα του led. Χρησιμοποιώντας τα 8 MSB του ADC ουσιαστικά μπορούμε να επιλέξουμε οποιοδήποτε duty cycle (χρησιμοποιούμε 8 bit Fast PWM) σε αντίθεση με το mode1 που διαλέγουμε κάποιες διακριτές τιμές για το duty cycle από τον πίνακα array.