Rješenje:

BRG16	BRGH	SPBRG	Zaokruženje SPBRG	Greška (%)
0	0	2,45	2	-15,06
			3	13,71
0	1	12,81	12	-6,21
			13	1,38
1	0	12,81	12	-6,21
			13	1,38
1	**1**	54,23	**54**	**-0,41**
			55	1,38

Najmanja greška iznosi -0,41% i postiže se za BRG16=1, BRGH=1, SPBRG=54.

Kod:

```
BRGH=1;             // podešavamo brzinu 54321 bps
BRG16=1;
SPBRGH=0;           //  54 je 0x0036
SPBRGL=0x36;
SYNC=0;             // koristimo asinhronu komunikaciju
                    // (nije neophodno, jer je default)
SPEN=1;             // uključujemo serijski port
CREN=1;             // uključujemo prijemnik
TXEN=1;             // uključujemo prijemnik
```

22. Odrediti najbolje vrijednosti BRGH, BRG16 i SPBRG, da bi se serijski port mikrokontrolera PIC16F1939 sa frekvencijom oscilatora od 16 MHz podesio na brzinu komunikacije od 14141 bps. Napisati kod kojim se modul za serijsku komunikaciju konfiguriše i postavlja na zadanu brzinu (asinhrona komunikacija, podatak od 8 bita, neparni paritet).

Odgovor:

BRG16	BRGH	SPBRG	Zaokruženje SPBRG	Greška (%)
0	0	16,68	16	-3,99
			17	1,78
0	1	69,72	69	-1,02
			70	0,40
1	0	69,72	69	-1,02
			70	0,40
1	**1**	281,87	281	-0,31
			282	**0,05**

Najmanja greška iznosi 0,05% i postiže se za BRG16=1, BRGH=1, SPBRG=282.

Kod:

```
BRGH=1;          // podešavamo brzinu 14141 bps
BRG16=1;
SPBRGH=1;        // 282 je 0x011A
SPBRGL=1A;
SYNC=0;          // koristimo asinhronu komunikaciju
                 // (nije neophodno, jer je default)
SPEN=1;          // uključujemo serijski port
CREN=1;          // uključujemo prijemnik
TXEN=1;          // uključujemo prijemnik
```

23. Odrediti najbolje vrijednosti BRGH, BRG16 i SPBRG, da bi se serijski port mikrokontrolera PIC16F1939 sa frekvencijom oscilatora od 12 MHz podesio na brzinu komunikacije od 11111 bps. Napisati kod kojim se modul za serijsku komunikaciju konfiguriše i postavlja na zadanu brzinu (asinhrona komunikacija, podatak od 8 bita, bez bita pariteta).

Odgovor:

BRG16	BRGH	SPBRG	Zaokruženje SPBRG	Greška (%)
0	0	15,875		
			15	5,470
			16	-0,734
0	1	66,501		
			66	0,747
			67	-0,734
1	0	66,501		
			66	0,747
			67	-0,734
1	**1**	269,003		
			269	0,001
			270	-0,368

Očigledno se najmanja greška (0,001%) postiže za BRG16=1, BRGH=1, SPBRG=269. Treba voditi računa da je SPBRG 16-bitni registar, koji se sastoji od registara SPBRGH i SPBRGL, a 16-bitna vrijednost se može upisati ili u registar SPBRG kao cjelinu, ili se u SPBRGH treba upisati vrijednost 1, a u SPBRGL vrijednost 13 (269=0x10D).

Kod za inicijalizaciju porta je dat u nastavku:

```
BRGH=1;                  // podešavamo brzinu 11111 bps
BRG16=1;
SPBRGH=1;
SPBRGL=13;
```

```
SYNC=0;                    // koristimo asinhronu komunikaciju
                           // (nije neophodno, jer je default)
SPEN=1;                    // uključujemo serijski port
CREN=1;                    // uključujemo prijemnik
TXEN=1;                    // uključujemo prijemnik
```

24. Odrediti najbolje vrijednosti BRGH, BRG16 i SPBRG, da bi se serijski port mikrokontrolera PIC16F1939 sa frekvencijom oscilatora od 10 MHz podesio na brzinu komunikacije od :
    - 22222 bps
    - 12488 bps
    - 12345 bps

    Napisati kod kojim se modul za serijsku komunikaciju konfiguriše i postavlja na zadanu brzinu (asinhrona komunikacija, podatak od 8 bita).

25. Napisati dio koda koji omogućava slanje vrijednosti koje se nalaze u varijabli deklarisanoj kao:

```
char lista[10];
```

    Pretpostaviti da je serijski port već konfigurisan, te da su vrijednosti pohranjene u varijablu.

    Odgovor:

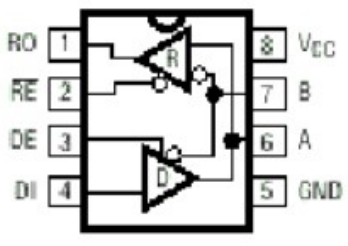
```
char k;
PORTCbits.RC5=1;              // uključujemo predaju na MAX485
TXEN=1;                       // uključujemo predajnik
for(k=0;k<10;k++){
   TXREG=lista[k];               // u TXREG ubacujemo znak
   while(!TXIF);           // čekamo da se oslobodi buffer
   TXIF=0;
}
TXEN=0;                       // isključujemo predajnik
PORTCbits.RC5=0;              // uključujemo prijem na MAX485
```

26. Napisati prekidnu rutinu koji omogućava prijem vrijednosti preko serijskog porta i njihovo pohranjivanje u varijablu deklarisanu kao:

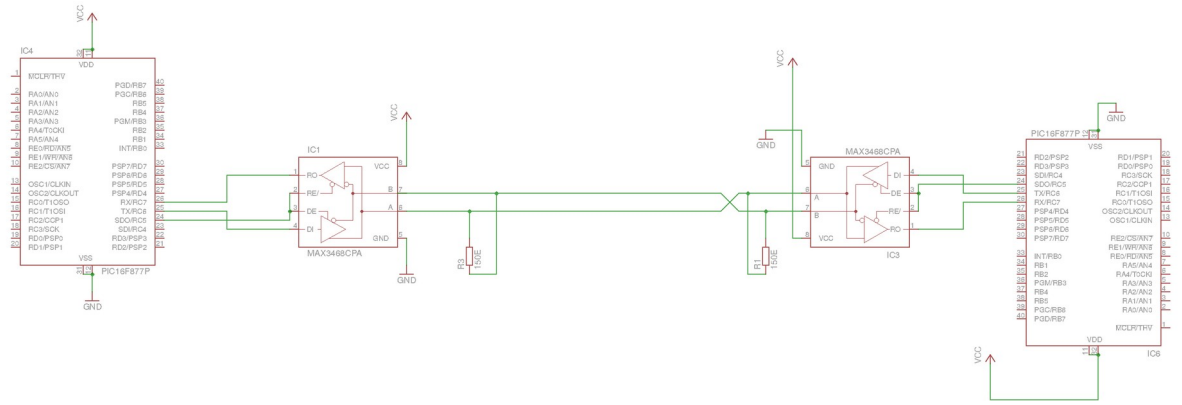
```
char lista[10];
```

    Kada se lista popuni, novi karakteri se pohranjuju od njenog početka. Pretpostaviti da je serijski port već konfigurisan, ali pojasniti šta je sve potrebno konfigurisati za aktiviranje prekida prijemnika serijskog porta.

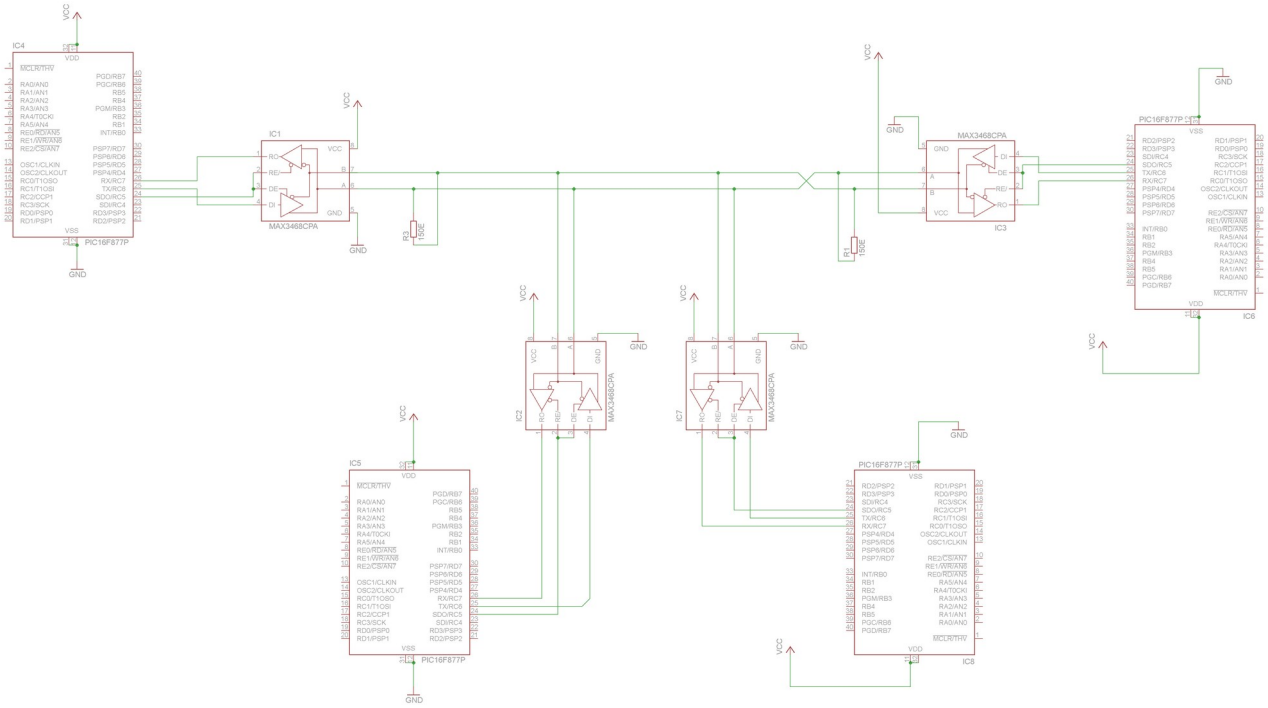
27. Nacrtati najjednostavniju šemu povezivanja, koja omogućava da se dva mikrokontrolera PIC16F1939 povežu korištenjem standarda RS-485. Raspored pinova za kolo MAX485 je prikazan na slici. Pinovi porta su: RC6 je TX, RC7 je RX.



Rješenje:



28. Nacrtati šemu povezivanja tri mikrokontrolera PIC16F1939 korištenjem RS-485 standarda.
29. Nacrtati šemu povezivanja četiri mikrokontrolera PIC16F1939 korištenjem RS-485 standarda. Pojasniti ulogu terminirajućih otpornika, te navesti njihovu vrijednost.



30. Napisati dio koda koji omogućava slanje vrijednosti iz varijable deklarisane kao

```
char lista[10];
```

preko serijskog porta. Pretpostaviti da je serijski port već konfigurisan, te da je sadržaj upisan u varijablu.

31. Popuniti tabelu:

Standard	Broj učesnika u komunikaciji	Maksimalna udaljenost	Vrsta signala i vrijednosti signala
RS-232	2	ca. 10 m	Bipolarni (-15V/-0,5V; 0,5V/15V)
RS-422	Max. 31	1200 m	Diferencijalni (0,5Vpp)
RS-485	Max. 31	1200 m	Diferencijalni (0,5Vpp)

Zadaci:

2. Potrebno je napisati programski kod u C-u za PIC16F1939, koji omogućava da se u okviru prekidne rutine prima karakter preko serijskog porta, te da se ovaj karakter pohranjuje u EEPROM kao podatak dužine 1 bajt, počevši od adrese 0x00. Svaka naredni primljeni se pohranjuje u narednu lokaciju EEPROM-a, a nakon što se popuni cijeli EEPROM, ponovo se vrijednosti pohranjuju od početne adrese, pri čemu se vrijednost na PORTB invertuje (sa 0x00 na 0xFF, odnosno sa 0xFF na 0x00 sljedeći put). Za podešenje serijskog porta koristiti kod iz 1.3.

   Rješenje:

   Da bi se aktivirao prekid na prijem karaktera preko serijskog porta, potrebno je postaviti flagove GIE, PEIE i RCIE. Kod je dat u nastavku:

```
#include <xc.h>
unsigned char znak, adresa=0;
void interrupt ser_int(void)
{
    if (RCIE && RCIF) {
        RCIF=0;
        znak=RCREG;                // čitamo znak primljen preko
                                   //  serijskog porta
        EEADRL=adresa++;           // postavljamo adresu lokacije
        EEDATL=znak;               // postavljamo podatak
        EECON2=0x55;                // obavezna sekvenca za upis
        EECON2=0xAA;                // je 0x55, 0xAA
        EECON1bits.WR=1;           // aktiviramo upisivanje
        while(!EEIF);               // čekamo kraj upisivanja
        EEIF=0;                     // resetujemo EEIF
        if(adresa==0) PORTB=~PORTB; // ukoliko je kraj EEPROM-a,
                                    //invertujemo ispis na PORTB
    }
}
```

```
void main(void) {
    TRISB=0;                        // PORTB je izlazni
    ANSELB=0;
    PORTB=0;                        // ispisujemo sve nule na PORTB
    WREN=1;                         // omogućavamo upis u EEPROM
    BRGH=1;                         // podešavamo brzinu 11111 bps
    BRG16=1;
    SPBRGH=1;
    SPBRGL=13;
    SYNC=0;                         // koristimo asinhronu komunikaciju
                                    // (nije neophodno, jer je default)
    RCIE=1;                         // aktiviramo prekid serijskog porta
    PEIE=1;
    GIE=1;
    SPEN=1;                         // uključujemo serijski port
    CREN=1;                         // uključujemo prijemnik
    while(1);
}
```

3. Potrebno je napisati programski kod u C-u za PIC16F1939, koji omogućava da se sa periodom od 1s očitava vijednost analognog napona dovedenog na pin RA0, te da se ova vrijednost pohranjuje u EEPROM kao podatak dužine 1 bajt, počevši od adrese 0x00. Svaka naredna očitana vrijednost se pohranjuje u narednu lokaciju EEPROM-a, a nakon što se popuni cijeli EEPROM, ponovo se vrijednosti pohranjuju od početne adrese. Za mjerenje vremena pri uzorkovanju koristiti prekid tajmera TMR0. Oscilator mikrokontrolera ima frekvenciju od 10 MHz.

   Rješenje:

   Obzirom da je f_{osc}=10 MHz, instrukcijski ciklus će trajati T_i=0,4 µs. Obzirom da modul TIMER0 generiše prekid na pretek (nakon 256 odbrojanih ciklusa), korištenjem preskalera (sa djeliteljem 1:256) možemo postići vrijeme $T=T_i \cdot 256 \cdot 256$ = 0,0262114 s. Da bi protekla 1 s, potrebno je odbrojati 1/T=38,14697 ≈ 38 preteka. Osim tajmera, potrebno je koristiti jedan analogni ulaz i EEPROM.

   Napomena: Na ovaj način se ne postiže tačno vrijeme od 1 s iz dva razloga:

   - **zaokruživanje broja preteka unosi grešku od 0,05878 µs po 1 s,**
   - **nije u obzir uzeto trajanje izvršavanja instrukcija u prekidnoj rutini.**

   Kod slijedi:

```
#include <xc.h>
char brojac=0, adresa=0, vrijednost;
void interrupt tc_int(void) {
```

```
    // U prekidnoj rutini cemo provjeravati da li se radi o prekidu TIMER0
    // i odbrojavati 38 prekida
    if (TMR0IE && TMR0IF) {
        TMR0IF=0;
        if (brojac<38)
            brojac++;
        else {
            brojac=0;
            // Ocitaj vrijednost na AN0
            ADCON0bits.ADGO=1;
            while(ADCON0bits.ADGO);      // Cekamo da se AD konverzija zavrsi
            vrijednost=ADRESH;           // Ocitavamo vrijednost iz ADC -
                                         // zanemaricemo najniza dva bita
            EEADRL=adresa++;             // postavljamo adresu lokacije
            EEDATL=vrijednost;           // postavljamo podatak
            EECON2=0x55;                 // obavezna sekvenca za upis
            EECON2=0xAA;                 // je 0x55, 0xAA
            EECON1bits.WR=1;             // aktiviramo upisivanje
            while(!EEIF);                // čekamo kraj upisivanja
            EEIF=0;                      // resetujemo EEIF
            if(adresa>255) adresa=0;     // ukoliko je kraj EEPROM-a, vracamo
                                         // se na prvu adresu
        }
    }
}
void init_analog(){
    // PORTA ce biti analogni
    TRISA=0xFF; // Nije neophodno, posto je to defaultna vrijednost
    ANSELA=0x01;
    PORTA=0;
    LATA=0;
    // Lijevo poravnanje
    ADCON1bits.ADFM=0;
    // Interni RC oscilator za ADC
    ADCON1bits.ADCS2=1;
    ADCON1bits.ADCS1=1;
    ADCON1bits.ADCS0=1;
    // Vss za Vref-, Vdd za Vref+
```

```
    ADCON1bits.ADNREF=0;
    ADCON1bits.ADPREF1=0;
    ADCON1bits.ADPREF0=0;
    // Ukljucivanje ADC
    ADCON0bits.ADON=1;
    // Izbor AN0
    ADCON0bits.CHS4=0;
    ADCON0bits.CHS3=0;
    ADCON0bits.CHS2=0;
    ADCON0bits.CHS1=0;
    ADCON0bits.CHS0=0;
}
void init_eeprom(void) {
    EEPGD=0;                        // odabiremo upis u EEPROM
    CFGS=0;                         // odabiremo EEPROM
    GIE=0;                          // deaktiviramo prekide
    EECON1bits.WREN=1;              // omogućavamo pisanje u EEPROM
}
void init_timer(void) {
    // Konfigurisemo brojanje instrukcijskih ciklusa, prescaler ukljucujemo
    // i postavljamo na 1:256
    TMR0CS=0;
    PSA=0;
    PS0=1;
    PS1=1;
    PS2=1;
    // Aktiviramo prekid TIMER0
    TMR0IF=0;                    // Brisemo TMR0IF (nije neophodno, jer ce na
                                 // pocetku svakako biti resetovan)
    TMR0IE=1;                    // Aktiviramo prekid TMR0
    GIE=1;                       // Aktiviramo prekide
}
void main(void) {
    init_analog();
    init_eeprom();
    init_timer;
    while(1);
}
```

4. Potrebno je napisati programski kod u C-u za PIC16F1939, koji omogućava da se na LED diodama povezanim na PORTB i spojenim sa zajedničkom anodom, prikaže cijeli broj koji odgovara vrijednosti napona dovedenoj na analogni ulaz AN0 (uz korištenje 8 bita veće težine). Očitavanje i prikaz vrijednosti realizirati u okviru prekidne rutine.

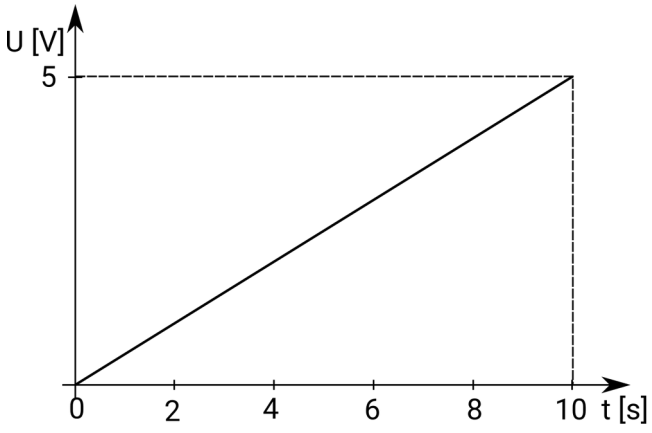
(5 poena)

Rješenje:

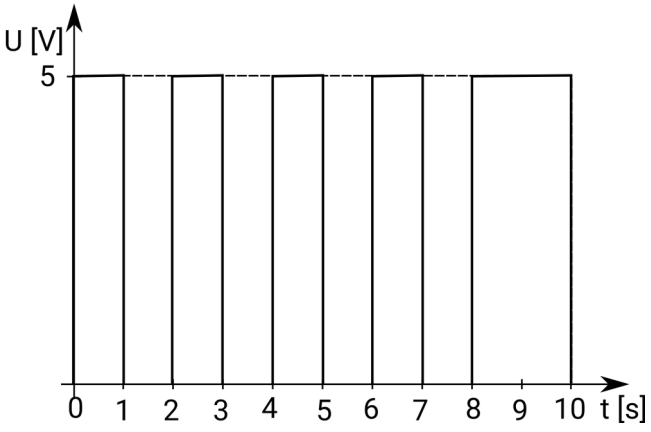
```
#include <xc.h>
void interrupt tc_int(void)
{
    // U prekidnoj rutini cemo izvrsavati AD konverziju i rezultat
    // prikazivati na PORTB
    if (ADIE && ADIF) {
        ADIF=0;
        PORTB=~ADRESH;      // koristeci bitwise invertovanje bita
        // moze i na drugi nacin:
        // PORTB=255-ADRESH;    // oduzimajuci ocitanu vrijednost od 0xFF
        // PORTB=255^ADRESH;    // izvodeci XOR izmedju 0xFF i ocitane vrijednosti
       ADGO=1;                 // ponovo pokrecemo AD konverziju
    }
    return;
}
void inicijalizacija(void) {
    // Na PORTB ce biti ispisivan rezultat AD konverzije
    TRISB=0x00;
    ANSELB=0x00;
    LATB=0x00;
    PORTB=0x00;
    // Aktiviramo prekid AD konvertora
     ADIF=0;                      // Brisemo TMR0IF (nije neophodno, jer ce na pocetku svakako biti
resetovan)
    ADIE=1;                   // Aktiviramo prekid TMR0
    PEIE_bit=1;               // Aktiviramo prekide od strane perifernih modula
    GIE=1;                    // Aktiviramo prekide
}
void init_analog(){
    // PORTA ce biti analogni
    TRISA=0xFF; // Nije neophodno, posto je to defaultna vrijednost
    ANSELA=0x01;
    PORTA=0;
    LATA=0;
    // Lijevo poravnanje
    ADCON1bits.ADFM=0;
    // Interni RC oscilator za ADC
    ADCON1bits.ADCS2=1;
    ADCON1bits.ADCS1=1;
    ADCON1bits.ADCS0=1;
    // Vss za Vref-, Vdd za Vref+
```

```
    ADCON1bits.ADNREF=0;
    ADCON1bits.ADPREF1=0;
    ADCON1bits.ADPREF0=0;
    // Ukljucivanje ADC
    ADCON0bits.ADON=1;
    // Izbor AN0
    ADCON0bits.CHS4=0;
    ADCON0bits.CHS3=0;
    ADCON0bits.CHS2=0;
    ADCON0bits.CHS1=0;
    ADCON0bits.CHS0=0;
    ADGO=1;    // da bi se desio prekid od strane AD konvertora, moramo pokrenuti AD konverziju
}
void main(void) {
    inicijalizacija();
    init_analog();
    while(1);
    return;
}
```

5. Na pinove RA0 i RB0 mikrokontrolera PIC16F1939 su dovedeni signali predstavljeni na slici 1 (signal se ne nastavlja nakon vremena od 10s). Mikrokontroler izvršava kod naveden u nastavku. Ispisati sve vrijednosti koje će biti postavljene na PORTD. Frekvencija oscilatora je 10MHz.



RA0



RB0

```
#include <xc.h>
unsigned char a, b=0;
void interrupt fun(void) {
    if (INTE && INTF) {
        INTF=0;
        GO=1;
        while(GO);
        PORTD=ADRESH;
    }
}
```

```
void main(void) {
    TRISD=0;
    ANSELD=0;
    INTEDG=1;
    PORTD=0;
    TRISB=0xFF;
    TRISA=0xFF;
    ANSELA=0x01;
    ADCON1bits.ADFM=1;
    ADCON1bits.ADCS2=1;
    ADCON1bits.ADCS1=1;
    ADCON1bits.ADCS0=1;
    ADCON1bits.ADNREF=0;
    ADCON1bits.ADPREF1=0;
    ADCON1bits.ADPREF0=0;
    ADCON0bits.ADON=1;
    ADCON0bits.CHS4=0;
    INTE=1;
    ADCON0bits.CHS3=0;
    ADCON0bits.CHS2=0;
    ADCON0bits.CHS1=0;
    ADCON0bits.CHS0=0;
    GIE=1;
    while(1);
}
```

Rješenje:

U kodu je aktiviran prekid na uzlaznu ivicu signala na pinu INT/RB0, tako da će se prekid desiti za t=0s, t=2s, t=4s, t=6s i t=8s. Čak i da se prekid za t=0s ne registrira, stanje na PORTD će inicijalno biti 0. Na ulaz RA0 se dovodi kontinualna vrijednost napona od 0-5V, u skladu sa linearnom funkcijom:

$$U_{RA0}=0{,}5\ t$$

Ovaj napon će se pretvoriti u cijeli broj, pri čemu se koristi 10-bitna rezolucija, uz desno poravnanje (ADFM=1). Radi toga su 2 bita najviše vrijednosti su u ADRESH, a 8 bita niže vrijednosti su u ADRESL. Obzirom da se na PORD ispisuje sadržaj ADRESH, samo dva bita najviše vrijednosti određuju vrijednost ispisanu na PORTD. Rezultat A/D konverzije se računa prema izrazu:

$$I=floor\left[\frac{U(t)}{5V}\cdot(2^{10}-1)\right]$$

Vrijednosti ispisa su predstavljene u tabeli.

t	0	2	4	6	8
$U_{RA0}(t)$	0	1	2	3	4
I	0=0x0000	204=0x00CC	409=0x0199	613=0x0265	818=0x0332
ADRESH	**0x00**	**0x00**	**0x01**	**0x02**	**0x03**
ADRESL	0x00	0xCC	0x99	0x65	0x32

6. Na bazi mikrokontrolera PIC16F1939 je realiziran sistem za registraciju srednje vrijednosti napona. Analogna vrijednost napona se dovodi na ulaz AN0, mjeri svakih 100ms, računa se srednja vrijednost ovog napona za jednu sekundu, te se ova srednja vrijednost pohranjuje u EEPROM i prikazuje na LED diodama povezanim sa zajedničkom anodom na PORTB. Nakon pokretanja sistema, prva vrijednost se pohranjuje na lokaciju 0x00 EEPROM-a, a svaka sljedeća srednja vrijednost se pohranjuje u narednu lokaciju EEPROM-a. Nakon što se EEPROM popuni, srednje vrijednosti se pohranjuju ponovo od lokacije 0x00. Napisati programski kod, koristiti 8-bitnu rezoluciju.

   Rješenje:

```
#include <xc.h>
double srednja=0;       // Srednja vrijednost napona
unsigned char brojac=0, sek=0; // Brojač za odbrojavanje 100ms, brojač za odbrojavanje 1s
unsigned char adresa=0;          // Adresa lokacije u EEPROM-u
char cijeli,decimale;   // Cijeli i decimalni dio vrijednosti napona

void init_digital() {
    // Srednja vrijednost se ispisuje na PORTB;
    TRISB=0;
    // LED su spojene sa zajedničkom anodom
    PORTB=~0;
}
void init_analog(){
    // PORTA ce biti analogni
    TRISA=0xFF; // Nije neophodno, posto je to defaultna vrijednost
    ANSELA=0x01;
    // Lijevo poravnanje
    ADCON1bits.ADFM=0;
    // Interni RC oscilator za ADC
    ADCON1bits.ADCS2=1;
    ADCON1bits.ADCS1=1;
    ADCON1bits.ADCS0=1;
    // Vss za Vref-, Vdd za Vref+
    ADCON1bits.ADNREF=0;
    ADCON1bits.ADPREF1=0;
```

```
    ADCON1bits.ADPREF0=0;
    // Ukljucivanje ADC
    ADCON0bits.ADON=1;
    // Izbor AN0
    ADCON0bits.CHS4=0;
    ADCON0bits.CHS3=0;
    ADCON0bits.CHS2=0;
    ADCON0bits.CHS1=0;
    ADCON0bits.CHS0=0;
}

void init_eeprom() {
    // Pristupamo EEPROM modulu
    EEPGD=0;
    CFGS=0;
    // Omogućavamo pisanje u EEPROM
    WREN=1;
}

void init_interrupt() {
    // Prekid TIMER0 koristimo za pokretanje AD konverzije svakih 100ms
    TMR0IE=1;
    TMR0IF=0;
    // Prekid AD konvertora koristimo za očitavanje rezultata AD konverzije
    ADIE=1;
    ADIF=0;
    PEIE=1;
    GIE=1;
}

void interrupt fun() {
    if(TMR0IE&&TMR0IF) {
        TMR0IF=0;
        if(brojac++>61) {       // Odbrojavanje 61 pretek za 100ms
            brojac=0;
            sek++;
            ADGO=1;             // Pokretanje AD konverzije
        }
        if(sek>=10) {            // Protekla 1s
            srednja/=10.0;        // Računanje srednje vrijednosti
            cijeli=(char)srednja;   // Cijeli dio srednje vrijednosti napona
            decimale=(char)((srednja-cijeli)*10);   // Decimalni dio
            EEDATL=(cijeli<<4)|decimale;    // Cijeli i decimalni u bajt
```

```
            EEADRL=adresa++;        // Adresa će se vratiti na 0 nakon 255
            EECON2=0x55;
            EECON2=0xAA;
            WR=1;
            while(!EEIF);
            EEIF=0;
            PORTB=~((cijeli<<4)|decimale);  // Ispis na LED (zaj. anoda)
            srednja=0;
            sek=0;
        }
    }
    if(ADIE&&ADIF) {
        ADIF=0;
        srednja+=ADRESH/255.0*5;              // Ažuriranje srednje vrijednosti
    }
}

void main(void) {
    init_digital();
    init_analog();
    init_eeprom();
    init_interrupt();
    while(1);
    return;
}
```

Pojašnjenje realizacije

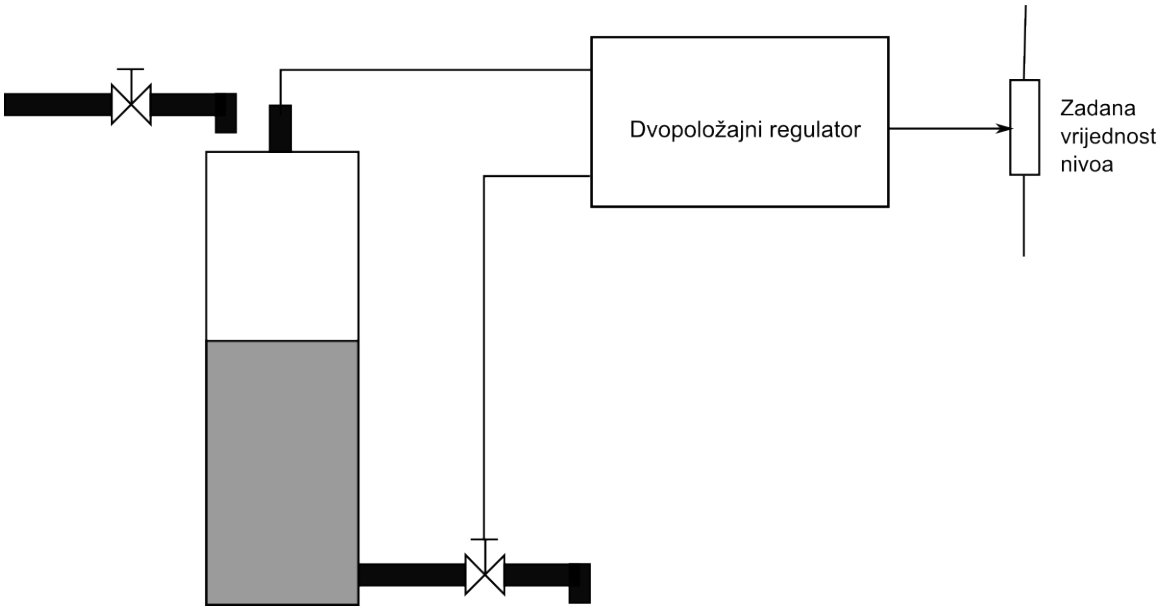
Rezultat AD konverzije se očitava kao 8-bitni cijeli broj, te se pretvara u vrijednost napona izraženu kao double. Obzirom da se zahtijeva upis ove srednje vrijednosti u lokaciju EEPROM-a, te ispis na PORTB, potrebno je na određeni način predstaviti ovu vrijednost kao char (1 bajt). Imajući u vidu da je potencijalno najviša izmjerena vrijednost napona 5V, a najniža 0V, za bilježenje cijelog dijela vrijednosti napona su dovoljna 4 bita (vrijednosti od 0 do 9). Isto tako, za bilježenje jedne decimalne cifre su dovoljna dodatna 4 bita (vrijednosti od 0 do 9). Radi toga je usvojeno da se u gornje 4 binarne cifre bilježi cijeli broj vrijednosti napona, a u donje 4 binarne cifre jedna decimala vrijednosti napona.

7. Sistem na bazi mikrokontrolera PIC16F1939 se koristi kao dvopoložajni regulator nivoa vode u rezervoaru. Za mjerenje nivoa se koristi ultrazvučni davač nivoa sa strujnim signalom u opsegu od 4-20 mA i linearnom karakteristikom. Pri tome struja od 4 mA odgovara nivou od 10 cm, a struja od 20 mA nivou od 100 cm. Strujni signal ultrazvučnog davača je potrebno pretvoriti u naponski signal u opsegu od 0-5 V, tako da struji od 20 mA

odgovara napon od 5 V. Ovaj napon se dovodi na analogni ulaz AN7 mikrokontrolera. Na analogni ulaz AN6 mikrokontrolera se dovodi napon sa potenciometra od 1kΩ, kojim se definira zadana vrijednost nivoa. A/D konvertor mikrokontrolera koristi vanjski izvor referentnog napona od 5 V. Na pin 6 porta D je priključen relej od 12 V, preko koga se vrši otvaranje ventila na odovodnom cjevovodu, tako da nivo vode u rezervoaru opada kada je ventil otvoren, a raste kada je ventil zatvoren. Potrebno je:

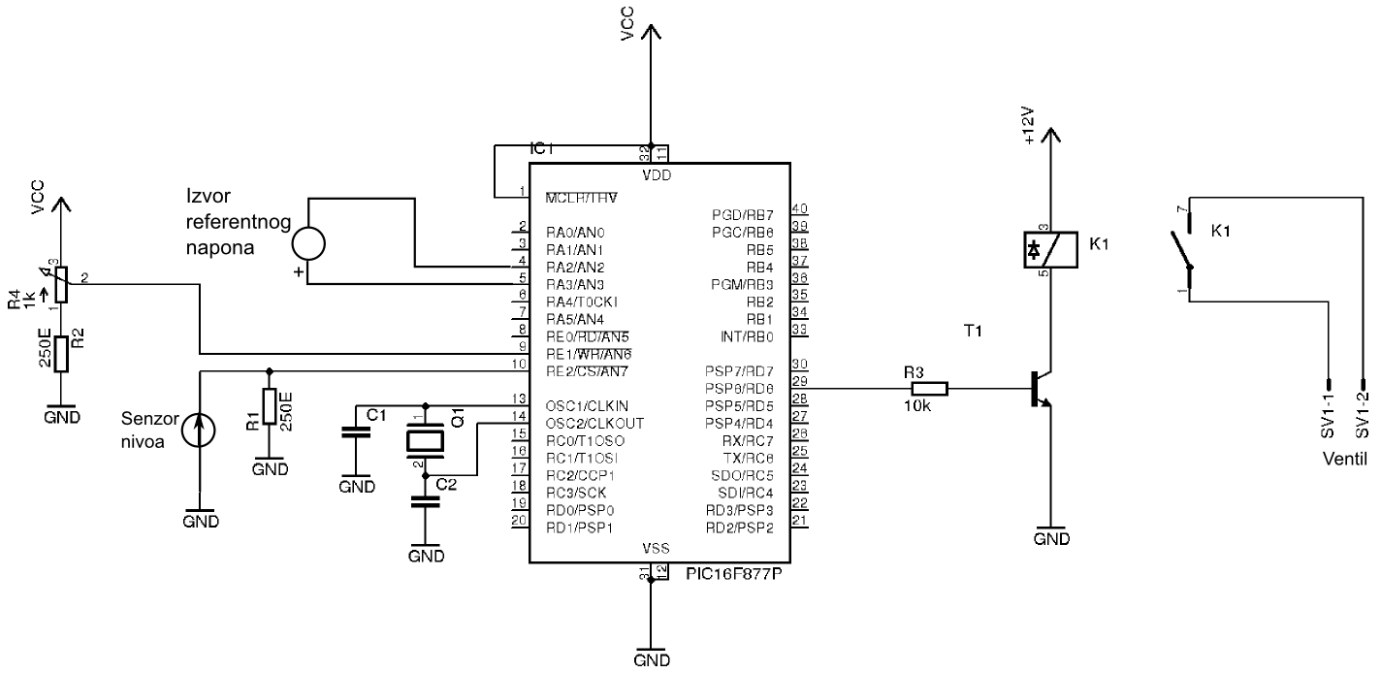
a) nacrtati električnu šemu povezivanja svih bitnih komponenata sistema regulacije nivoa;

b) napisati program u C-u koji realizira dvopoložajni regulator za ovaj sistem.



Rješenje:

a)



b)

```
#include <xc.h>
void init_analog(){
    // PORTE ce biti analogni
    TRISE=0xFF; // Nije neophodno, posto je to defaultna vrijednost
    ANSELE=0x06;
    // Lijevo poravnanje
    ADCON1bits.ADFM=0;
    // Interni RC oscilator za ADC
    ADCON1bits.ADCS2=1;
    ADCON1bits.ADCS1=1;
    ADCON1bits.ADCS0=1;
    // Vref- i Vref+ iz vanjskog izvora
    ADCON1bits.ADNREF=1;
    ADCON1bits.ADPREF1=1;
    ADCON1bits.ADPREF0=0;
    // Ukljucivanje ADC
    ADCON0bits.ADON=1;
    // Izbor AN6
    ADCON0bits.CHS4=0;
    ADCON0bits.CHS3=0;
    ADCON0bits.CHS2=1;
    ADCON0bits.CHS1=1;
    ADCON0bits.CHS0=0;
}
void inicijalizacija(void){
      TRISDbits.TRISD6=0;            // pin RD6 je izlazni
      PORTDbits.RD6=0;               // Ventil je zatvoren
      init_analog();
}
void main(void){
      char nivo, zadnivo;             // Varijable za pohranu nivoa i zad. nivoa
      inicijalizacija();
      while(1){
            CHS0=1;                   // Izbor kanala AN7 - mjerenje nivoa
            GO=1;                     // Pokretanje A/D konverzije
            while(GO);                // Čekanje da se konverzija završi
            nivo=ADRESH;              // Očitavanje izmjerene vrijednosti nivoa
```

```
            CHS0=0;                    // Izbor kanala AN6 – zadani nivo
            GO=1;
            while(GO);
            zadnivo=ADRESH;            // Očitavanje zadane vrijednosti nivoa
            if(nivo>zadnivo)           // Otvaranje ventila kada je izmjereni
                  PORTDbits.RD6=1;     // nivo veći od zadanog
            else
                  PORTDbits.RD6=0;     // Zatvaranje ventila u suprotnom slučaju
      }
}
```