Mladen Vučić

UGRADBENI RAČUNALNI SUSTAVI
Upute za laboratorijske vježbe
2. ciklus

Zagreb, 2007.

Sadržaj

1 Uvod

Ovaj tekst sadrži materijale za drugu laboratorijsku vježbu kolegija Ugradbeni računalni sustavi studija FER-2. Vježba je namijenjene upoznavanju sa sklopovskim i programskim alatima i, koliko to kratkoća vremena dozvoljava, stjecanju određenih vještina potrebnih za razvoj ugrađenih računalnih sustava. Gradivo obuhvaća rad s mikrokontrolerima, pri čemu je kao primjer odabrana familija mikrokontrolera 8051 [1–4], te programski paket uVision proizvođača Keil Elektronik GmbH [5]. Težište rada stavljeno je na vezu programa i sklopovlja tokom razvoja uređaja. U tom smislu ovi materijali se nadovezuju na literaturu [6–8].

Poglavlja 2, 3 i 4 obuhvaćaju upoznavanje sa sklopovljem koje će se na vježbi koristiti, te upoznavanje s alatima za razvoj programske podrške. Poglavlja od 5 do 14 pokrivaju samostalan rad kod kuće, te rad u laboratoriju u kojem se programska podrška razvija i ispituje na stvarnom sklopovlju.

Upute za rad dane su pod pretpostavkom da se koristi programski paket uVision2. Danas su dostupne i novije inačice ovog programa. Obzirom da se njihovo korisničko sučelje vrlo malo razlikuje od sučelja uVision2, upute za rad se mogu koristiti i za njih. Na mjestima na kojima postoji značajnija razlika, to je posebno napomenuto.

2 Upoznavanje sa sklopovljem

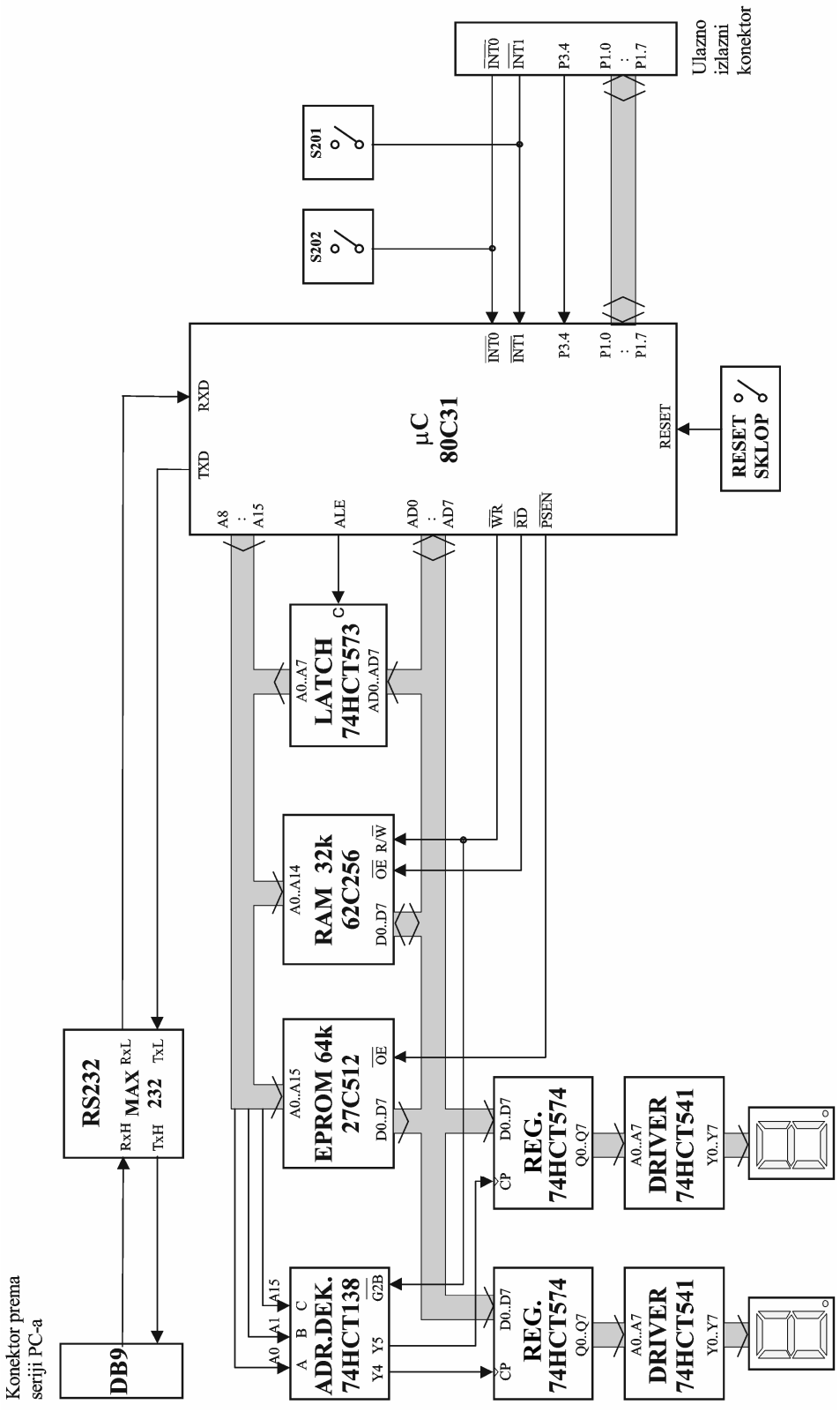
2.1 Računalni sustav

Slika 2.1 prikazuje blokovsku shemu sklopovlja računalnog sustava s mikrokontrolerom 8031. 8031 je inačica mikrokontrolera 8051, bez unutrašnje programske memorije. Vanjska programska memorija u ovom slučaju ima kapacitet 64 kilobajta (27C512). U sustav je dodana i vanjska podatkovna memorija kapaciteta 32 kilobajta (62C256).

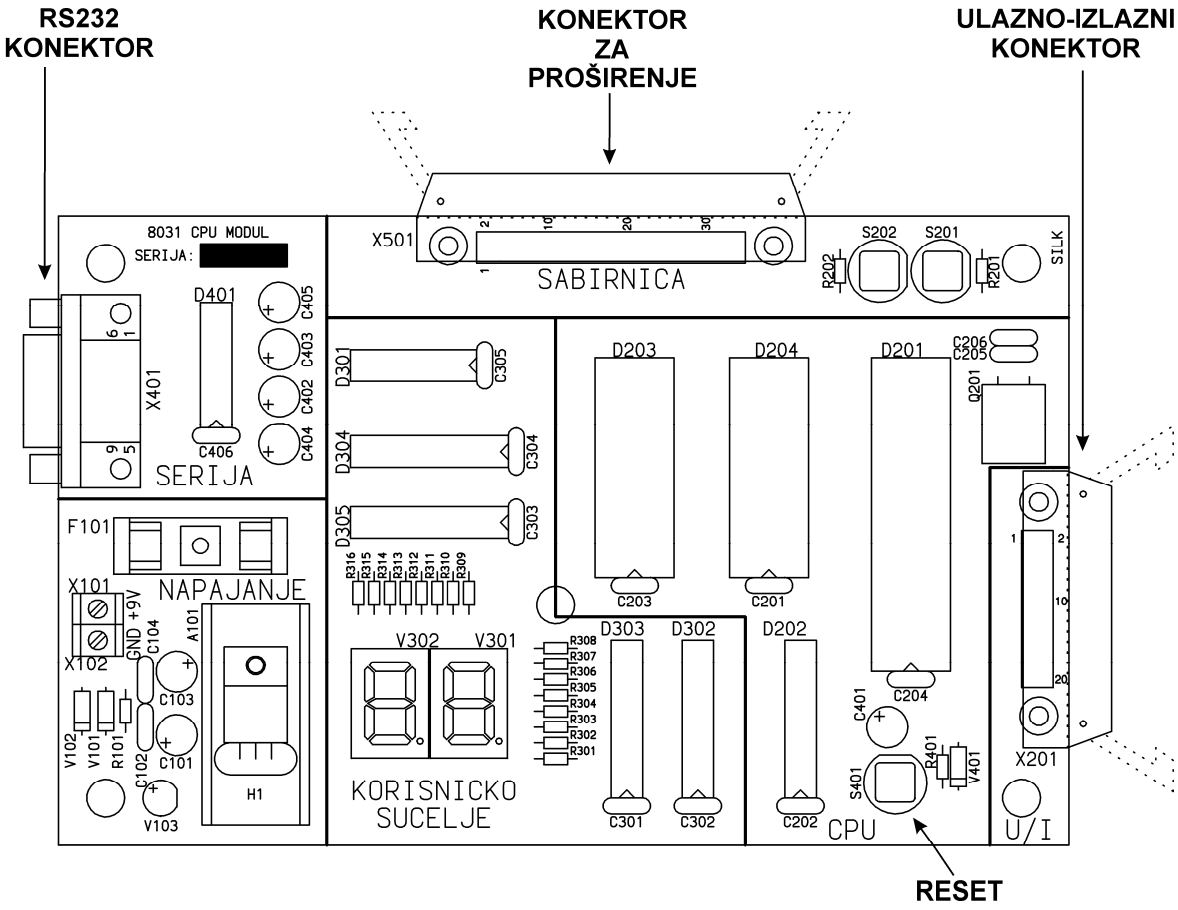
Kao pokazna jedinica služe dvije znamenke sedam-segmentnih pokaznika sa svjetlećim diodama. Prikaz znaka na pokazniku ostvaruje se upisom odgovarajućeg bajta u registar 74HCT574, koji služi kao memorija. Registri se "vide" u podatkovnom memorijskom prostoru, tj. pristupa im se na isti način kao i vanjskoj podatkovnoj memoriji. Dekodiranje je napravljeno pomoću "3 na 8" dekodera 74HCT138. Sklopovi 74HCT541 služe kao pojačala koja mogu dati dovoljno veliku struju za pogon pokaznika. Sustav ima priključak za serijsku vezu (RS232) te se može povezati s terminalom ili osobnim računalom koje emulira terminal. Pretvorba napona TTL razine na približno ±10V koje zahtijeva standard serijske veze ostvarena je integriranim sklopom MAX232. Veza s "vanjskim svijetom" ostvarena je preko skupa priključaka P1 koji je spojen na konektor, zajedno s priključkom 4 skupa P3 te priključcima za vanjske prekide $\overline{\text{INT0}}$ i $\overline{\text{INT1}}$ koji također pripadaju skupu P3.

Električne sheme sustava prikazane su slikama 4.1, 4.2, 4.3, 4.4 i 4.5, danim u poglavlju 4.

Sustav je fizički napravljen na dvoslojnoj štampanoj pločici veličine 10x16 cm čiji raspored elemenata prikazuje slika 2.2. Na istoj pločici nalazi se i sklopovlje za napajanje, te konektor za proširenje sustava koji sadrži adresne linije, A0 do A15, podatkovne linije, D0 do D7, komunikacijske linije RXD i TXD, upravljačke linije ALE, $\overline{\text{PSEN}}$, $\overline{\text{RD}}$ i $\overline{\text{WR}}$ te linije za napajanje +5V i GND.



Slika 2.1 Blokovska shema računalnog sustava s mikrokontrolerom 8031.



Slika 2.2 Raspored elemenata štampane pločice računalnog sustava.

2.2 EPROM-emulator

Tokom razvoja sustava, a posebno tokom razvoja programske podrške, umjesto EPROM-a u njegovo podnožje spaja se EPROM-emulator. Time se znatno ubrzava postupak uhodavanja programske podrške. Naime, za vađenje iz podnožja, brisanje, te programiranje EPROM-a potrebno je 15 minuta. Ako se radi s FLASH EPROM-om, potrebno je 1 do 2 minute. Postupak punjenja EPROM emulatora *(download)* traje svega nekoliko sekundi.

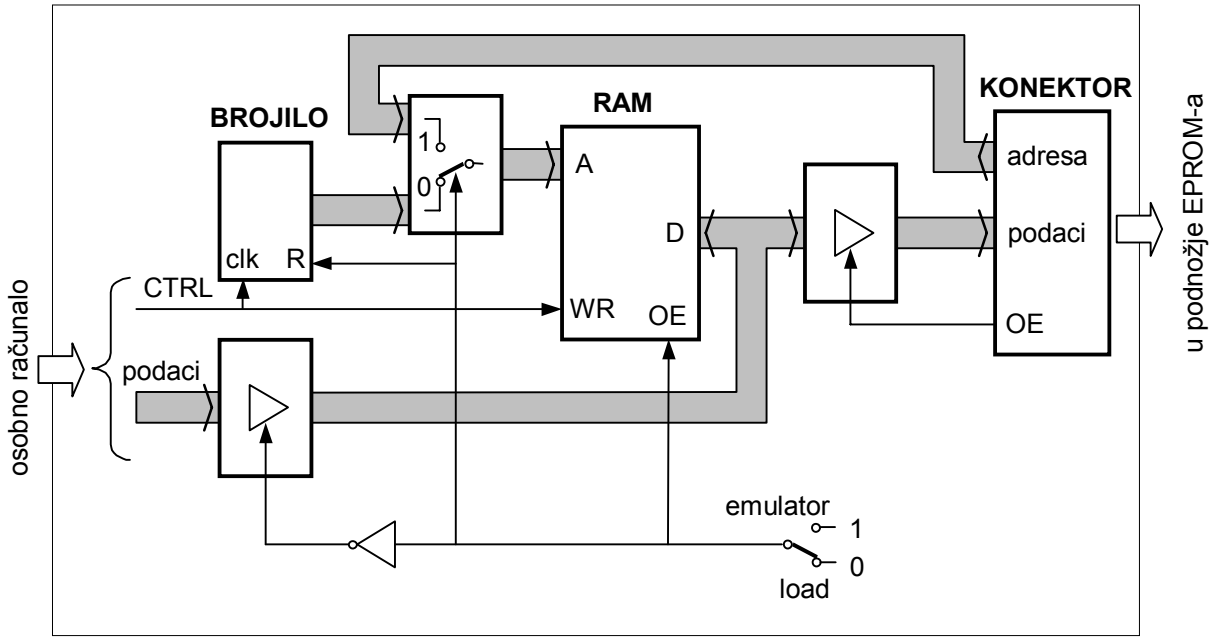
Na tržištu postoji čitav niz različitih EPROM-emulatora. Svi su oni slični po funkciji. Slika 2.3 prikazuje blokovsku shemu tipičnog EPROM-emulatora, a jedna izvedba prikazanog sklopa bit će korištena na laboratorijskim vježbama.

Emulator ima dva načina rada; upis *(load)* i čitanje tj. emulaciju *(emulation)*. Za vrijeme upisa, podaci se prenose s osobnog računala i pohranjuju u RAM memoriji. U našem slučaju za prijenos se koristi paralelna veza i to 8 bitova podatka (bajt) i jedan upravljački signal (CTRL). Tokom upisa sklopka je u položaju **load**. Podatkovne linije memorije su u stanju visoke impedancije, a podaci iz osobnog računala propuštaju se na podatkovne linije memorije. Adrese se generiraju pomoću brojila, tj. signal iz brojila dovodi se na adresne linije memorije. Na početku upisa stanje brojila je 0000h jer je tokom prijašnjeg stanja emuliranja bilo cijelo vrijeme u stanju "reset". Osobno računalo na paralelno sučelje postavlja prvi bajt koji se želi upisati u RAM i generira impuls na

upravljačkoj liniji. Taj impuls omogućuje upis podatka u memoriju (WR) na adresu 0, te povećava sadržaj brojila za jedan, tj. priprema adresu 1. Računalo sada na paralelno sučelje postavlja drugi podatak, daje upravljački impuls i ponavlja postupak dok se ne prenesu svi podaci.

Nakon što su podaci preneseni u RAM, sklopka se postavlja u položaj **emulator**. Veza prema osobnom računalu se prekida, a adrese se sada dovode s podnožja u koje je uključen emulator. Iz memorije se čitaju podaci koji se prenose na podnožje kada je aktivna linija OE.

Tokom emulacije, brojila se nalaze u stanju "reset" da bi kod slijedećeg upisivanja počelo s generiranjem adresa od nule.



Slika 2.3 Pojednostavljena blokovska shema EPROM-emulatora.

U našem slučaju, za upisivanje *(download)* podataka s osobnog računala potrebno je prebaciti sklopku u položaj **load** (svijetli crvena LED dioda), te na tipkovnici računala otipkati komandu

copy imeprog.bin lpt1: /b

gdje je imeprog.bin naziv datoteke koja sadrži binarni kod programa mikrokontrolera. Nakon što su podaci preneseni, sklopku treba prebaciti u položaj **emulator**.

2.3 Termostat i tipkovnica

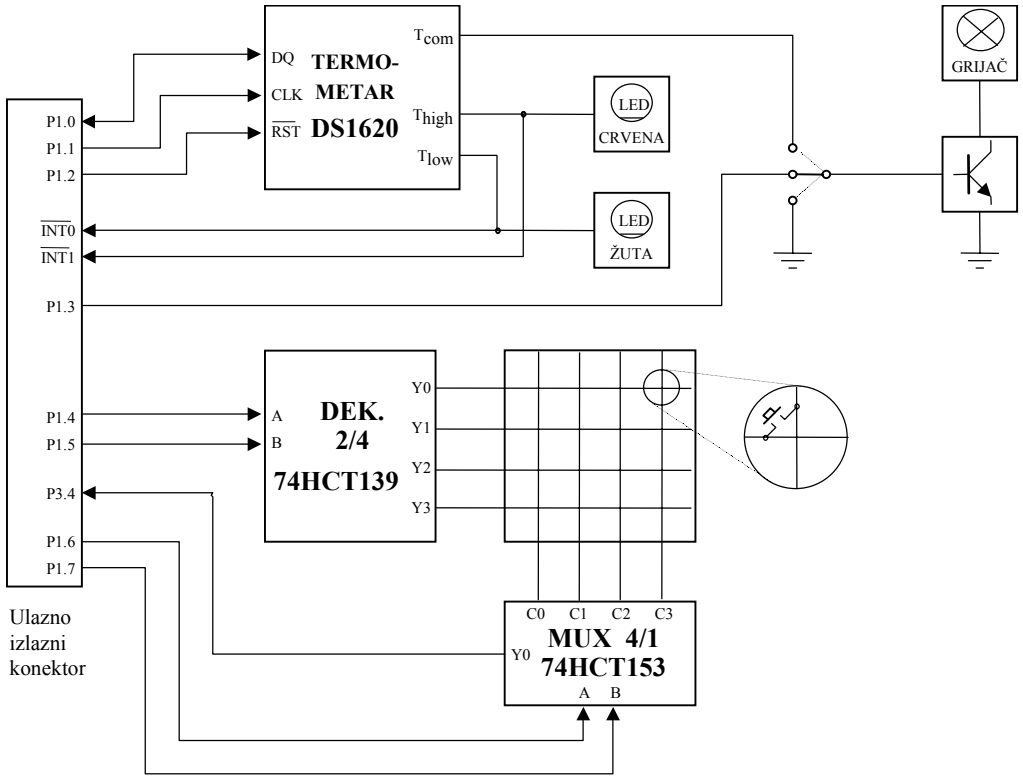
Slika 2.4 prikazuje blokovsku shemu sklopovlja termometra i tipkovnice, koji se nalaze na zajedničkoj štampanoj pločici. Pločica se s računalnim sustavom povezuje preko ulazno-izlaznog konektora koji sadrži sve priključke skupa P1, priključak 4 skupa P3 te linije za vanjski prekid $\overline{\text{INT0}}$ i $\overline{\text{INT1}}$ koje su također dio skupa P3 (sekundarne funkcije).

Gornji dio blokovske sheme prikazuje spoj termometra. Termometar je integrirani sklop u DIL8 kućištu. Sadrži 3 linije za serijsku komunikaciju s računalnim sustavom. U našem slučaju te linije su spojene s najniža tri bita supa priključaka P1, tj. P1.0, P1.1 i P1.3. Linija DQ služi za prijenos podataka, linija CLK za prijenos takta, a linija $\overline{\text{RST}}$ za upravljanje prijenosom podataka.

Tri izlazne linije T_{com}, T_{high} i T_{low} služe za komunikaciju s vanjskim svijetom kada se sklop koristi kao samostalan termostat. U našem slučaju linije T_{high} i T_{low} spojene su na linije za vanjski prekid mikrokontrolera. Ove linije neće se koristiti tokom ove vježbe.

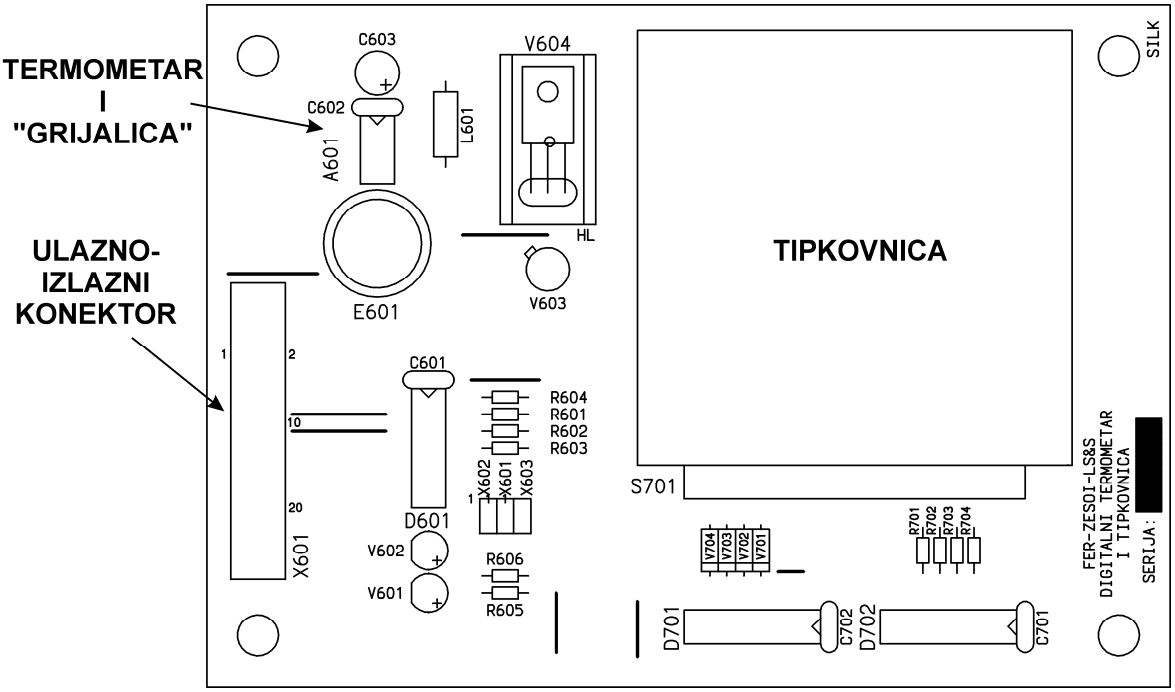
Na priključak mikrokontrolera P1.3, spojena je sklopka koja uključuje grijalicu. U našem slučaju grijalica je žarulja, smještena ispod termometra. Žarulja se uključuje upisom nule, a isključuje upisom jedinice na P1.3.

Električne sheme sustava prikazane su slikama 4.6 i 4.7, danim u poglavlju 4.



Slika 2.4 Blokovska shema termometra i tipkovnice.

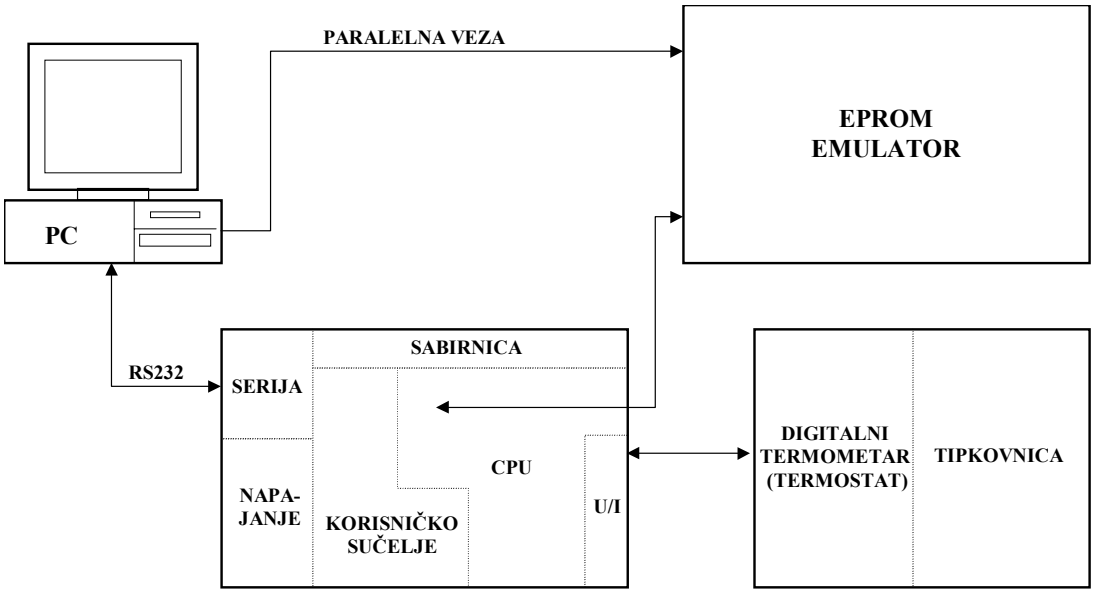
Slika 2.5 prikazuje fizički izgled štampane pločice termometra i tipkovnice. Sklopovlje koje će biti korišteno na ovoj vježbi smješteno je u gornjem lijevom uglu (A601, E601, itd.).



Slika 2.5 Fizički izgled štampane pločice termometra i tipkovnice.

2.4 Spajanje pojedinih dijelova sustava

Slika 2.6 prikazuje način spajanja svih modula koji će biti korišteni tokom ove vježbe. Module je potrebno spojiti prema danoj shemi, ali ne priključivati napajanje do početka rada s maketom.



Slika 2.6 Blokovska shema spajanja modula.

3 Upoznavanje s programskim alatima

3.1 Programski paket za razvoj programske podrške

3.1.1 Sadržaj paketa

Za razvoj programske podrške mikrokontrolera familije 8051 na vježbama će se koristiti poznati programski paket **uVision** proizvođača **Keil Elektronik GmbH** [5]. Paket je namijenjen radu na osobnom računalu, pod operacijskim sustavom Windows. Paket sadrži slijedeće module odnosno programe:

- **A51** - prevodilac za asembler
- **C51** - prevodilac za jezik C *(C compiler)*
- **BL51** - program za povezivanje relokatibilnog koda *(linker)* te njegovo smještanje u memoriju *(loader)*
- **Debugger/Simulator** - skup programa za simulaciju rada mikrokontrolera i pripadajuće programske podrške
- **OHS51** - program za kreiranje takozvanog Intel-HEX formata programa
- **LIB51** - program za rad s bibliotekama *(libraries)*

Svi spomenuti programi pozivaju se iz zajedničkog korisničkog sučelja pod nazivom **uVision**.

Izvorni kod programa piše se u asembleru ili C jeziku. Pritom se može koristiti bilo koji program za obradu teksta, ili, što je preporučljivo, editor koji se nalazi u sklopu korisničkog sučelja uVision. Programu pisanom u asembleru dodaje se nastavak ".a51", a programu pisanom u C jeziku nastavak ".c". Na primjer, imena programa mogu biti:

program1.a51
program2.c

Prevođenje programa pisanog u asembleru obavlja prevodilac *(assembler)* **A51**. Kao rezultat uspješnog prevođenja nastaju dvije datoteke

program1.obj - datoteka koja sadrži relokatibilni kod
program1.lst - datoteka s ispisom rezultata prevođenja

Za prevođenje izvornog koda pisanog u C jeziku, koristi se prevodilac *(compiler)* **C51**. Slično kao i kod asemblera, datoteke koje nastaju nakon uspješnog prevođenja imaju oblik

program2.obj - datoteka koja sadrži relokatibilni kod
program2.lst - datoteka s ispisom rezultata prevođenja

U slučaju kad program sadrži greške u sintaksi, u datoteci s nastavkom ".lst" piše gdje se te greške nalaze.

Nakon prevođenja, relokatibilni moduli (.obj) spajaju se i dodjeljuju im se stalne adrese. Taj posao obavlja program za povezivanje odnosno punjenje *(linker/loader)* pod nazivom **BL51**. Rezultat povezivanja odnosno punjenja je datoteka koja sadrži izvršni program, te datoteka koja sadrži apsolutne adrese varijabli i kodnih segmenata. Ova datoteka ima ime oblika

program1.m51

Izvršni program može se testirati u simulatoru gdje je moguće provjeravati funkcionalnost programa, pokretati ga korak po korak, pratiti stanje registara i varijabli itd.

Izvršni program dobiven povezivanjem i punjenjem nije zapisan u binarnom formatu već ga je potrebno najprije pretvoriti u takozvani Intel-HEX format, a nakon toga iz Intel-HEX formata u binarni format. Pretvorbu u Intel-HEX format obavlja program **OHS51** koji je dio paketa Keil uVision. Pretvorbu u binarni format u našem slučaju obavljat ćemo izvan okruženja paketa uVision, pomoću programa **hexbin**. Kao rezultat nastat će datoteka čije ime ima oblik

program.bin

koju možemo učitati u EPROM-emulator.

Opisane radnje, odnosno pokretanje programa većinom se odvijaju "u pozadini", tj. nisu izravno vidljive u korisničkom sučelju. Ipak, radi potpunijeg razumijevanja rada programskog paketa uVision, potrebno je poznavati opisani postupak.

3.1.2 Posebnosti prevodioca Keil - C51 vezane uz podatke

C prevodilac programskog paketa Keil razvijen je za familiju mikrokontrolera 8051, a zove se C51. On podržava specijalne tipove podataka i omogućava direktno pridruživanje pojedinih varijabli točno određenim memorijskim prostorima.

Pored tipova varijabli predviđenih ANSI C standardom, kod Keil-ovog prevodioca postoje i dodatni tipovi koji su potrebni zbog same arhitekture mikrokontrolera 8051. Tipovi varijabli koje podržava prevodilac dani su u tablici 3.1. Pored ovih tipova, postoje i specijalni tipovi varijabli za pristup registrima posebne namjene, takozvanom SFR prostoru. Ovi tipovi dani su u tablici 3.2.

Tablica 3.1Tipovi varijabli koje podržava prevodilac C51.

Tip varijable	**Veličina**	**Područje vrijednosti**
bit	1 bit	0 ili 1
signed char	1bajt	-128 do +127
unsigned char	1 bajt	0 do 255
signed int	2 bajta	-32768 do +32767
unsigned int	2 bajta	0 do 65535
signed long	4 bajta	-214783648 do +2147483647
unsigned long	4 bajta	0 do 4294967295
float	4 bajta	±1.176E-38 do ±3.40E-38
pointer	1-3 bajta	ovisi o adresi objekta

Tablica 3.2Tipovi varijabli za pristup registrima posebne namjene.

Tip varijable	**Veličina**	**Područje vrijednosti**
sbit	1 bit	0 ili 1
sfr	1bajt	0 do 255
sfr16	2 bajta	0 do 65535

Sustavi s mikrokontrolerom sadrže razne memorijske prostore, ovisno o tome koliko memorije postoji, da li su one vanjske ili unutrašnje itd. Tipovi memorije u kojima se mogu nalaziti varijable dani su u tablici 3.3.

Tablica 3.3Tipovi memorije u kojima se mogu nalaziti varijable.

Tip memorije	**Opis**
data	Direktno adresabilni unutrašnji RAM (donjih 128 bajtova unutrašnjeg RAM-a).
bdata	Bit adresabilni unutrašnji RAM (adrese od 20h do 2Fh)
idata	Indirektno adresabilni unutrašnji RAM (gornjih 128 bajtova unutrašnjeg RAM-a)
pdata	Vanjski RAM od 256 bajtova (dostupan preko MOVX @Ri)
xdata	Vanjski RAM od 64 kilobajta (dostupan preko MOVX @A+DPTR)
code	Vanjska programska memorija (ROM). Može se samo čitati pomoću MOVC @A+DPTR

Programer kroz deklaraciju varijable određuje u kojem memorijskom prostoru će se varijabla nalaziti. Pritom se rukovodi vremenom pristupa pojedinoj memoriji, kao i njenom veličinom. Najkraće vrijeme potrebno je za pisanje ili čitanje iz unutrašnjeg RAM-a, jer se on može direktno adresirati. Indirektno adresiranje gornjih 128 bajtova unutrašnjeg RAM-a traje duže jer je potrebno inicijalizirati registar koji sadrži adresu. Najsporiji je pristup vanjskom RAM-u.

Deklaracija varijable u općem slučaju sadrži:

- tip varijable
- memorijski prostor u kojem se nalazi varijabla
- ime varijable

kao na primjer:

```
/* Varijabla pero tipa unsigned integer u vanjskom RAM-u. */
unsigned int xdata pero;
/* Varijabla a12 tipa signed character, koja se nalazi negdje unutar donjih
128 bajtova unutrašnjeg RAM-a. */
signed char data a12;
```

Ako se u deklaraciji ne navede tip memorije, prevodilac će odabrati memorijski prostor ovisno o zadanom memorijskom modelu koji može biti **SMALL**, **COMPACT** ili **LARGE**. U SMALL modelu, varijable kojima nije zadan tip memorije stavljaju se u DATA prostor, u COMPACT modelu stavljaju se u IDATA prostor, a u LARGE modelu stavljaju se u XDATA prostor.

Deklaracija registara posebne namjene (SFR) ima oblik:

```
sfr P0 = 0x80;      /* Deklaracija SFR registra. */
sbit TI = 0x99;     /* Deklaracija bita u SFR registru. */
```

U slučaju deklaracije SFR registra broj iza znaka jednakosti predstavlja fizičku adresu deklariranog registra. Kod deklaracije bita u SFR registru broj iza znaka jednakosti sadrži adresu registra i adresu bita u registru. Svi bit adresabilni registri u SFR prostoru leže na "okruglim" adresama, kao na primjer 0x80, 0x88, 0x90, 0x98 itd. Adresa odgovarajućeg bita u registru dobiva se zbrajanjem adrese registra i mjesta bita. U našem slučaju bit TI dio je registra SCON čija je adresa 0x98. Bit TI nalazi se na mjestu 1 (koje odgovara 2^1),

pa se u deklaraciji nalazi adresa 0x98+1=0x99. Uočiti da ovo nije adresa bita već samo način zapisa. Ovakav način zapisa je nepregledan, ali još uvijek se koristi. Preglednije, bit u SFR registru može se deklarirati kao

```
sfr P0 = 0x80;
sbit P0_1 = P0^1;
```

ili, jednostavno kao

```
sbit P0_1 = 0x80^1;
```

Keil C51 prevodilac podržava memorijski određene *(memory specific)* i generičke *(generic)* pokazivače *(pointers)*. Memorijski određene pokazivače koristimo kad znamo na koji memorijski prostor oni pokazuju i kad se taj prostor ne mijenja tokom izvođenja programa. Primjer deklaracije određenog pokazivača je:

```
char data *pok1;      /* Troši 1 bajt jer sadrži adresu u data RAM-u */
char xdata *pok2;     /* Troši 2 bajta jer sadrži adresu u xdata RAM-u */
```

U ovom slučaju pok1 je pokazivač na karakter koji se nalazi u data prostoru. Izraz

```
pok1=&pero;
```

dozvoljen je samo ako je pero karakter u data prostoru, a ne npr. u xdata prostoru. Ovi pokazivači troše malo mjesta u memoriji, 1 bajt za data, odnosno 2 bajta za xdata prostor.

Generički pokazivači mogu pokazivati na bilo koji memorijski prostor. Primjer je

```
char *pok3;           /* Troši 3 bajta*/
```

U ovom slučaju pok3 je pokazivač na karakter koji se može nalaziti u bilo kojem memorijskom prostoru. Kad napišemo izraz

```
pok3=&pero;
```

nije važno kako je varijabla pero deklarirana, kao data, xdata itd. Ovi pokazivači troše 3 bajta. Prvi bajt govori o kojem je memorijskom prostoru riječ, a preostala dva bajta predstavljaju adresu u tom prostoru.

Sam pokazivač se nalazi u prostoru koji određuje memorijski model (small, compact ili large). Ipak, i sam pokazivač može se staviti u željeni memorijski prostor. Deklaracija pokazivača koji se nalazi u "data" prostoru, a pokazuje na integer u "xdata" prostoru ima oblik:

```
data int xdata *pok4;
```

ili

```
int xdata *data pok4;
```

Ukoliko je neka varijabla promjenjivog sadržaja, ona se deklarira kao varijabla tipa **volatile**, što dolazi od engleske riječi brisivo, promjenjivo. To je potrebno u situacijama u kojima više uzastopnih čitanja iste memorijske lokacije daje različite rezultate. Na primjer, ako se na lokaciji 0xFFFF u vanjskom memorijskom prostoru nalazi 8-bitni A/D pretvornik, svako novo čitanje s ove lokacije dat će u pravilu drugačiji podatak. Varijablu za pristup ovoj lokaciji treba deklarirati kao

```
volatile unsigned char xdata *AD_converter=0xFFFF;
```

Ukoliko se varijabla ne deklarira kao volatile, neki optimizatori koda koji su sastavni dio C prevodilaca mogu izbaciti dio koda s obrazloženjem da višestruko čitanje iste varijable daje uvijek isti rezultat, te pročitati varijablu samo jednom.

3.1.3 Posebnosti prevodioca Keil - C51 vezane uz funkcije

Prevodilac Keil - C51 podržava dva tipa funkcija: **reentrant** funkcije i **non-reentrant** funkcije. Pored toga, u funkcijama je moguće odrediti koja registarska banka se koristi, a postoji i učinkovit način pisanja funkcija koje obrađuju prekide.

Reentrant funkcije

Takozvane "reentrant" funkcije su funkcije koje se smiju pozivati rekurzivno. Također, takve funkcije smiju se pozivati istovremeno iz glavnog programa i funkcija koje obrađuju prekide.

Prilikom poziva funkcije, u funkciju se prenose parametri, tj. njeni argumenti, a funkcija vraća neku vrijednost (ukoliko se ne radi o funkciji tipa void). Tokom izvršavanja funkcije, koriste se neke memorijske lokacije kao lokalne varijable. Kod reentrant funkcija lokalne varijable se pohranjuju na simuliranom podatkovnom stogu. Taj stog treba razlikovati od sklopovskog stoga u kojem je pohranjena adresa za povrat iz potprograma. U "small" modelu, podatkovni pokazivač stoga zove se ?C_IBP.

Non-reentrant funkcije

Kod funkcija koje nisu tipa reentrant, lokalne varijable se pohranjuju u registrima ili na nepromjenjivim memorijskim lokacijama. Zato takve funkcije nije dozvoljeno pozvati istovremeno iz glavnog programa i funkcija koje obrađuju prekid. Naime, ako se to učini, obje pozvane funkcije (tj. ista funkcija pozvana dva puta) će pisati lokalne varijable po istim lokacijama memorije i zato dati pogrešne rezultate.

Prijenos argumenata u funkciju i iz funkcije

Prilikom poziva funkcije, argumenti se u funkciju prenose preko registara, kao što prikazuje tablica 3.4. Tako na primjer funkcija

funk1 (int a)

prenosi argument "a" preko registara R6 i R7. Funkcija

funk2 (int b, int c, int *d)

prenosi argument "b" preko registara R6 i R7, argument "c" preko R4 i R5, a argument "d" preko R1, R2 i R3. Funkcija

funk3 (float g, char h)

prenosi argument "g" preko registara R4 do R7, a argument "h" preko memorijske lokacije u RAM-u, a ne preko registra R5. Preko memorijskih lokacija se prenose i parametri od četvrtog nadalje.

Izlazna vrijednost funkcije prenosi se preko registara kao što to prikazuje tablica 3.5.

Tablica 3.4Prijenos ulaznih argumenata funkcije preko registara.

	char, 1-bajtni pok.	**int, 2-bajtni pok.**	**long, float**	**generički pok.**
1. parametar	R7	R6, R7	R4 do R7	R1, R2, R3
2. parametar	R5	R4, R5	R4 do R7	R1, R2, R3
3. parametar	R3	R2, R3	-	R1, R2, R3

Tablica 3.5Prijenos izlazne vrijednosti funkcije preko registara.

Izlazna vrijednost	**Registar**	**Opis**
bit	Carry-Flag	
char	R7	
int	R6, R7	MSB u R6, LSB u R7
long	R4 do R7	MSB u R4, LSB u R7
float	R4 do R7	32 bita IEEE format (eksponent i predznak u R7)
pointer	R1, R2, R3	selektor memorijskog prostore u R3, MSB adrese u R2, LSB adrese u R1

Određivanje registarske banke

U unutrašnjoj programskoj memoriji mikrokontrolera 8051 nalaze se četiri registarske banke, B0 do B3. Prevodilac će za pojedine operacije koristiti banku B0. To je predefinirano, ali se može i promijeniti pomoću odgovarajuće komande ili opcije.

Kad se na mikrokontroleru "paralelno" obavljaju dva zadatka, npr. glavni program i prekidni program, pojedini zadaci ne smiju "kvariti" posao drugih zadataka. Na primjer, kad se počne izvršavati funkcija koja obrađuje prekid, ona ne smije promijeniti sadržaj trenutno aktivne registarske banke, jer će glavni program nakon prekidne funkcije nastaviti svoje izvršavanje na mjestu na kojem je prekinut.

Da se sačuva sadržaj tekuće banke, moraju se na stog pohraniti sadržaji svih registara koji će se koristiti u prekidnoj funkciji. To se izvodi pomoću instrukcije PUSH. Na kraju prekidne rutine sadržaj registara mora se ponovo sa stoga prenijeti u registre, što se izvodi pomoću instrukcije POP.

Drugi način čuvanja sadržaja registara je promjena aktivne registarske banke. Promjena banke unutar funkcije zadaje se u definiciji funkcije, kao na primjer:

void funk10 (void) using 3

gdje funkcija funk10 radi s bankom B3.

Iako se promjena banke najčešće koristi kod prekidnih funkcija, može se koristiti i u drugim slučajevima. Pri tome treba biti oprezan jer funkcije koje se pozivaju iz drugih funkcija moraju raditi s istim registarskim bankama kao i funkcije iz koje su pozvane. Generalno, promjena banke nije uvijek korisna kad funkcija vraća podatak već je pogodnija kod funkcija tipa "void".

Prekidne funkcije

Prekidne funkcije definiraju se kao

void f_prekid (void) interrupt n using m

Atribut "using m" se može, ali ne mora staviti u definiciju funkcije. Atribut "interrupt n" znači da se radi o prekidnoj funkciji, pri čemu je "n" oznaka izvora prekida kao što prikazuje tablica 3.6.

Tablica 3.6Određivanje izvora prekida i adresa prekidnih funkcija.

n	Izvor prekida	Adresa prekidne rutine
0	vanjski prekid 0	0003h
1	brojilo T0	000Bh
2	vanjski prekid 1	0013h
3	brojilo T1	001Bh
4	serijsko sučelje	0023h

Nakon što se dogodi prekid, počne izvršavanje prekidne funkcije. Na početku funkcije prevodilac će umetnuti programski odsječak koji sprema sadržaj svih registara koji će biti korišteni u funkciji (ACC, B, DPH, DPL i PSW). Ako ne postoji atribut "using", spremit će se i cijela registarska banka (ukoliko je potrebno, tj. ako se u funkciji koristi). Ako postoji "using", sadržaj registara iz tekuće banke neće se spremati, čime se dobiva na brzini izvođenja programa. Sadržaj registara sprema se na stog (PUSH). Na kraju prekidne rutine podaci se uzimaju sa stoga (POP).

3.1.4 Primjer programa u jeziku C

Kod pisanja programa u jeziku C, potrebno je osim zahtjeva koji proizlaze iz standarda, uvažiti i neke druge zahtjeve. Kao prvo, program je potrebno pisati pregledno, s jasno istaknutim modulima, te odgovarajuće uvučenim blokovima. Nadalje, potrebno je komentarima pojasniti pojedine programske odsječke.

U daljnjem tekstu dan je trivijalan primjer programa da bi se ilustrirao spomenuti način pisanja. Program također služi i kao podsjetnik na redoslijed pisanja pojedinih dijelova koda.

```
/*=========================================================================*/
/*                              IME PROGRAMA                               */
/*=========================================================================*/

/* Komentar o tome što program radi i ostalim vaznim stvarima.            */
/* OVAJ PROGRAM PISAN JE ZA SINGLE CHIP IZVEDBU SUSTAVA S uC 8051.        */
/* Sluzi samo za podsjetnik kako treba izgledati struktura C programa.    */
/* Program cita podatak sa skupa prikljucaka P1, pribraja mu BROJ1 i      */
/* pise rezultat na skup P2                                               */

/*=========================================================================*/
/*         Prototipovi funkcija i deklaracije SFR registara                */
/*=========================================================================*/

#include  <reg51.h>           /* deklaracije SFR-ova */

unsigned char zbroji(unsigned char, unsigned char); /* funk.za zbrajanje */

/* Ako je funkcija u drugom file-u treba pisati ovako */
/* extern unsigned char zbroji(unsigned char, unsigned char);  */

/*=========================================================================*/
/*                              Konstante                                  */
/*=========================================================================*/

#define BROJ1       0x02            /*  */

/*=========================================================================*/
/*                         Globalne varijable                              */
/*=========================================================================*/

/* Ovdje idu deklaracije globalnih varijabli, ako ih ima. */

/*=========================================================================*/
/*                                  MAIN                                   */
/*=========================================================================*/

main()
{

/* deklaracije varijabli */

   unsigned char data pero;        /* komentar s opisom varijable */
   unsigned char data izlaz;       /* komentar s opisom varijable */

/* inicijalizacije */

   /* npr. inicijalizacije serije, brojila, prekida itd.,
      vec prema tome sto se koristi                      */

/* PETLJA KOJA SE NEPRESTANO IZVRSAVA */

   while(1)
      {
      /* dio programa koji neprestano cita podatak sa skupa P1, dodaje mu
         BROJ1 i pise rezultat na skup P2 */
      pero=P1;
      izlaz=zbroji(pero,BROJ1);
      P2=izlaz;
      }
}
```

```
/*=========================================================================*/
/*                           Funkcija zbroji                               */
/*=========================================================================*/

/* Komentar o funkciji, sto radi itd. */
/* Ova funkcija je samo primjer jer se ovako jednostavne stvari ne
   stavljaju u funkcije */

unsigned char zbroji(unsigned char a, unsigned char b)
{
   /* deklaracija lokalnih varijabli */
   unsigned char data c;
   c=a+b;
   return (c);
}
```

3.2 Program za emulaciju terminala

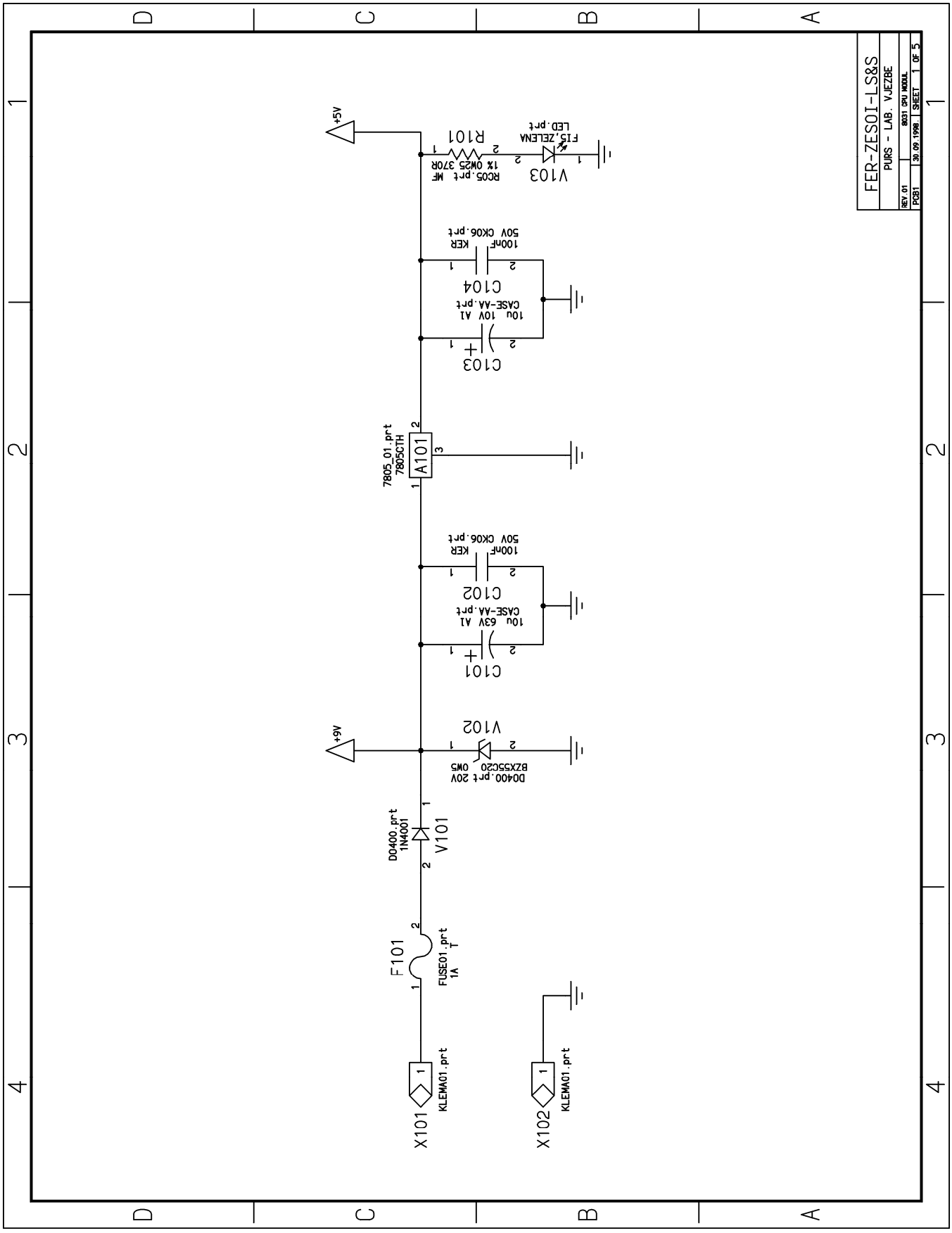
Da bi osobno računalo radilo kao terminal, potrebno je pokrenuti program za emulaciju terminala. U našem slučaju koristit ćemo program koji postoji u Windows okruženju, a čija se prečica *(shortcut)* naziva

Hyper Terminal

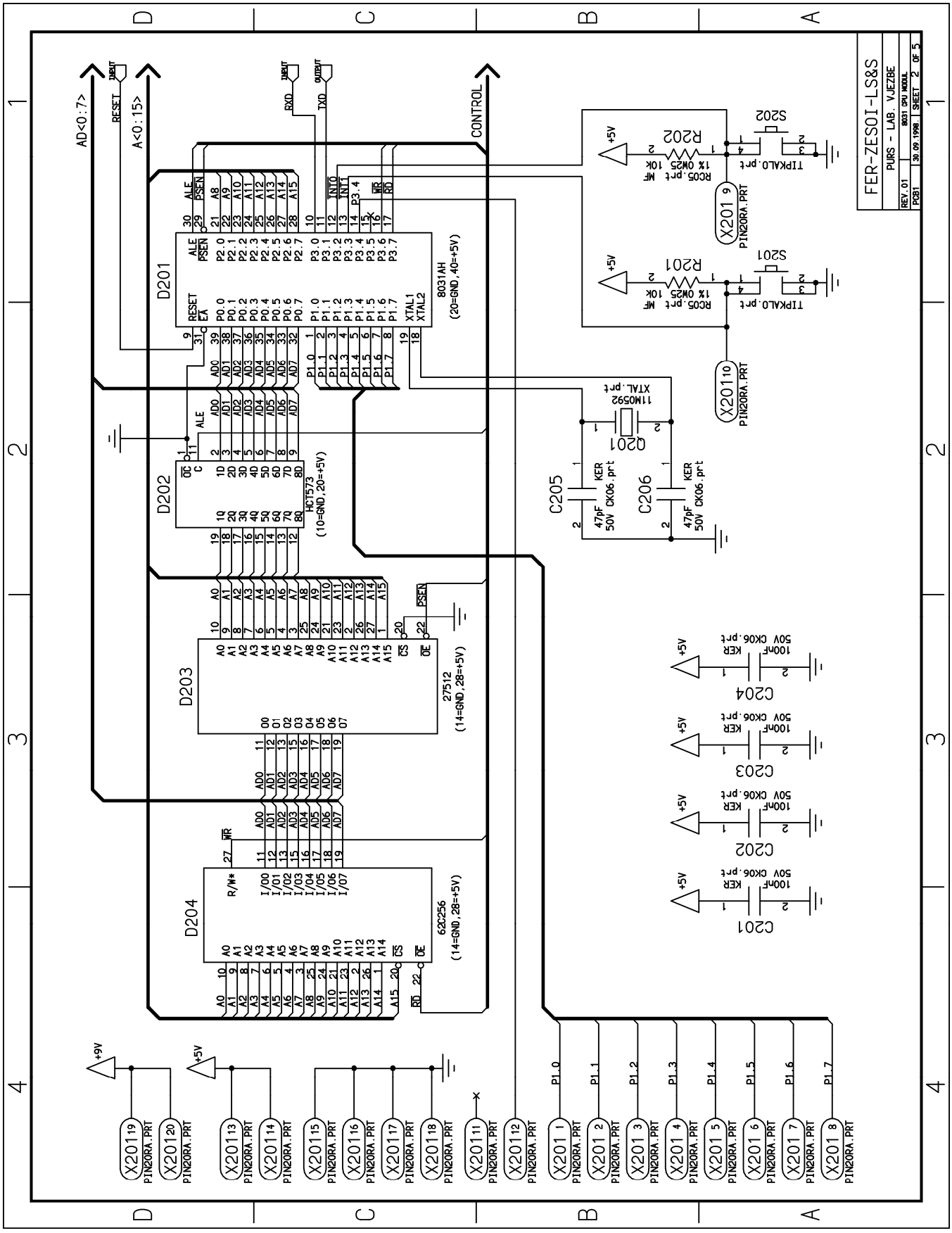
4 Električne sheme sklopovlja

U nastavku teksta dane su električne sheme sklopovlja koje se koristi na vježbi. Sheme obuhvaćaju

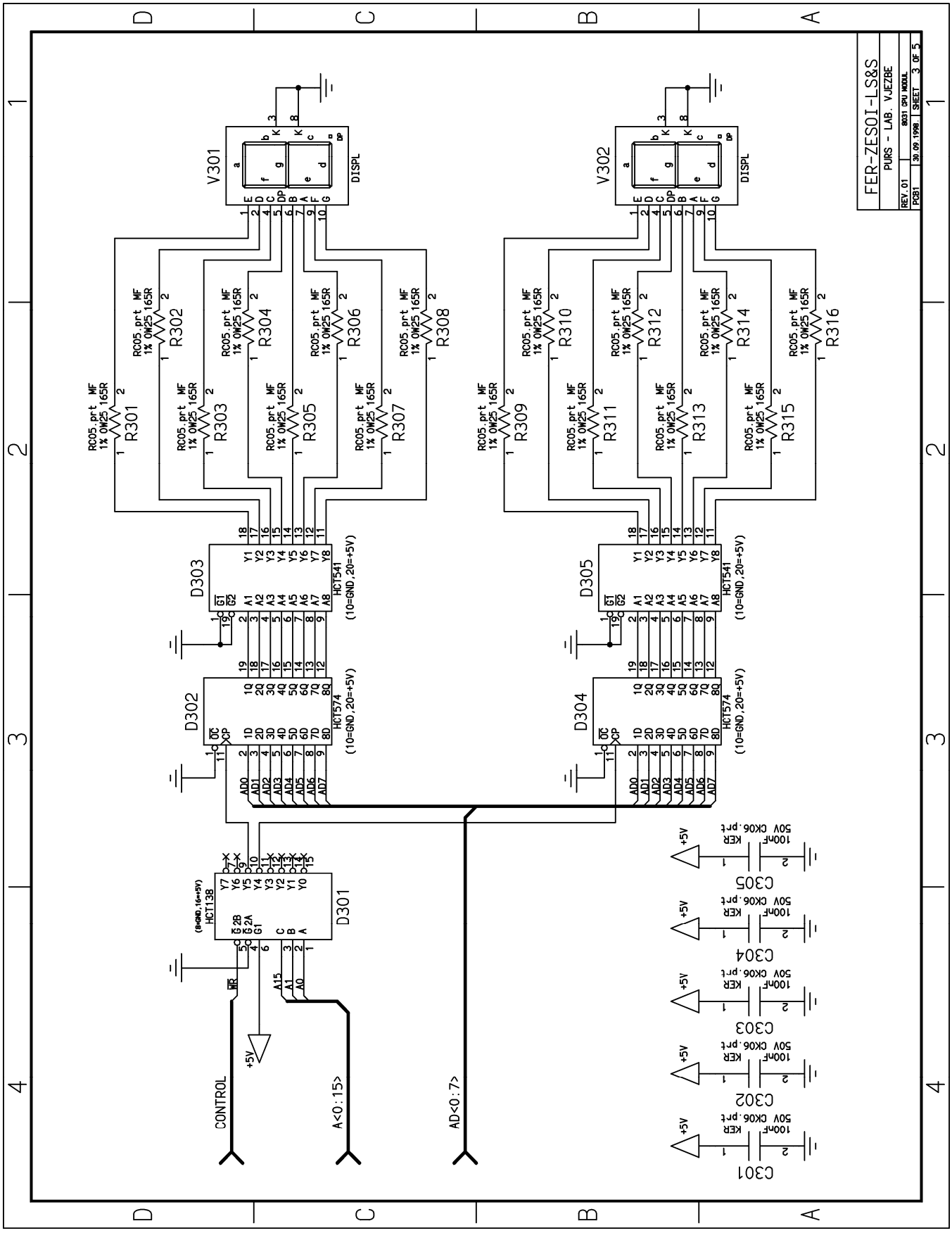
- računalni sustav s mikrokontrolerom 8031, slike 4.1, 4.2, 4.3, 4.4 i 4.5
- termostat, slika 4.6
- tipkovnicu, slika 4.7



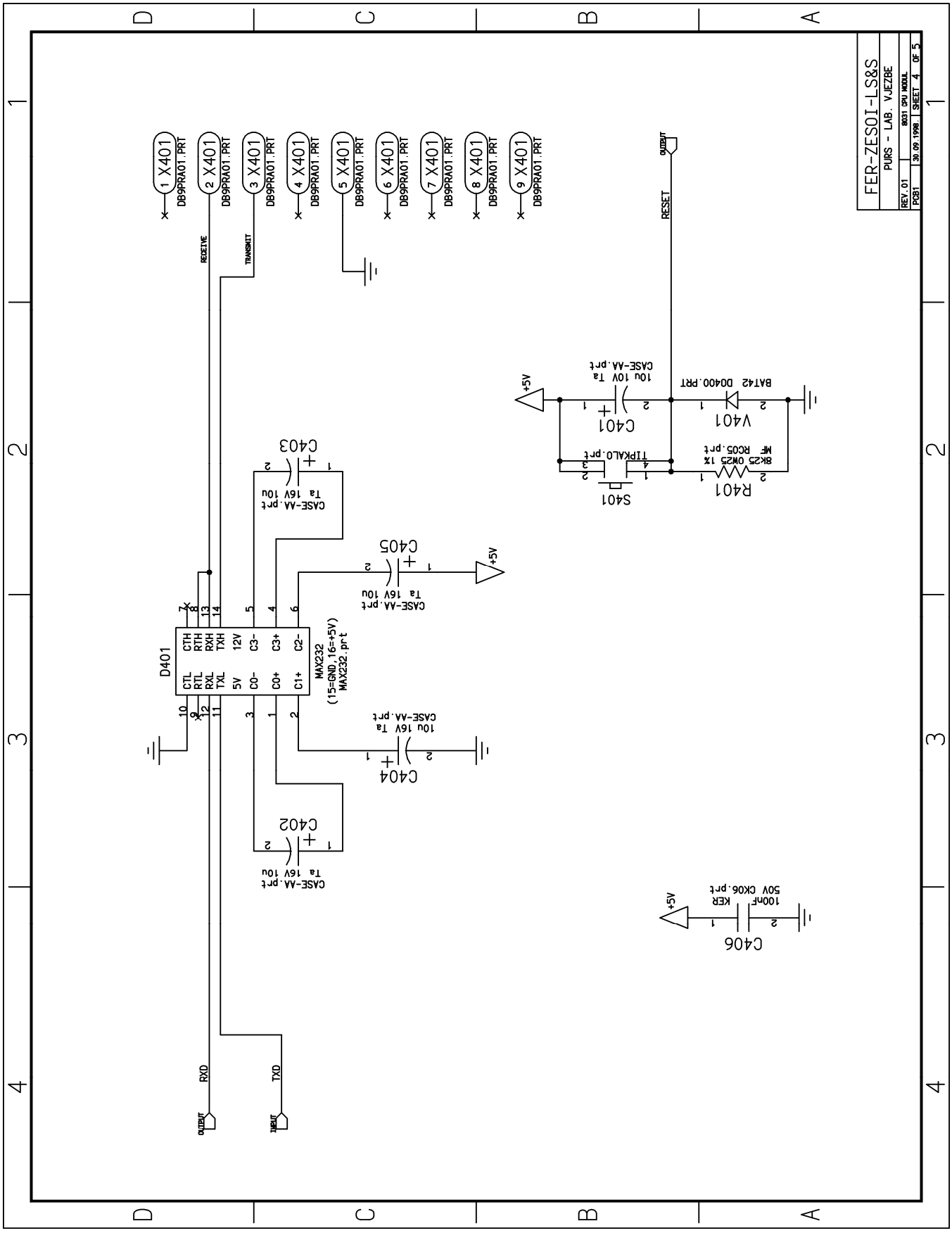
Slika 4.1 Električna shema računalnog sustava s mikrokontrolerom 8031, strana 1/5.



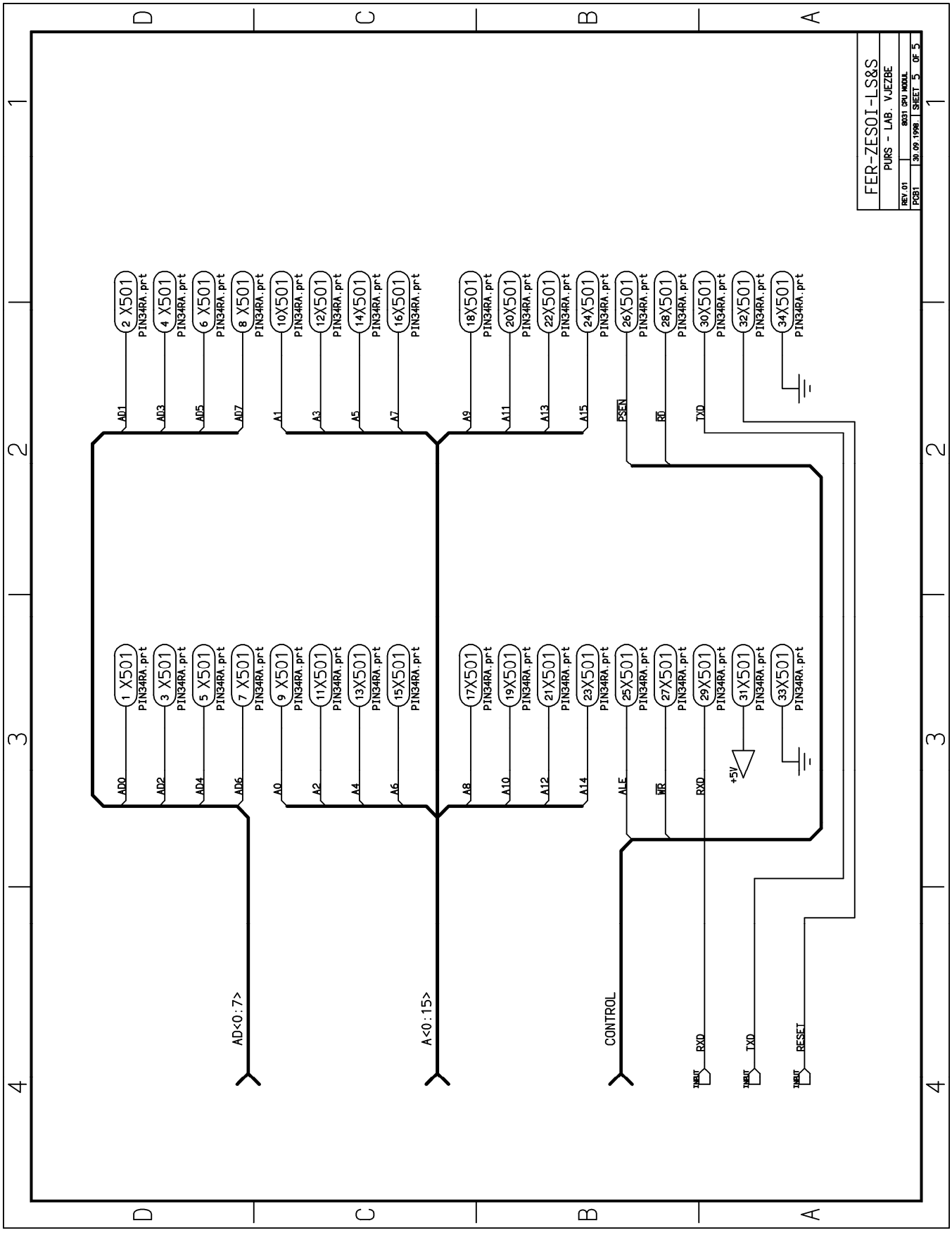
Slika 4.2 Električna shema računalnog sustava s mikrokontrolerom 8031, strana 2/5.



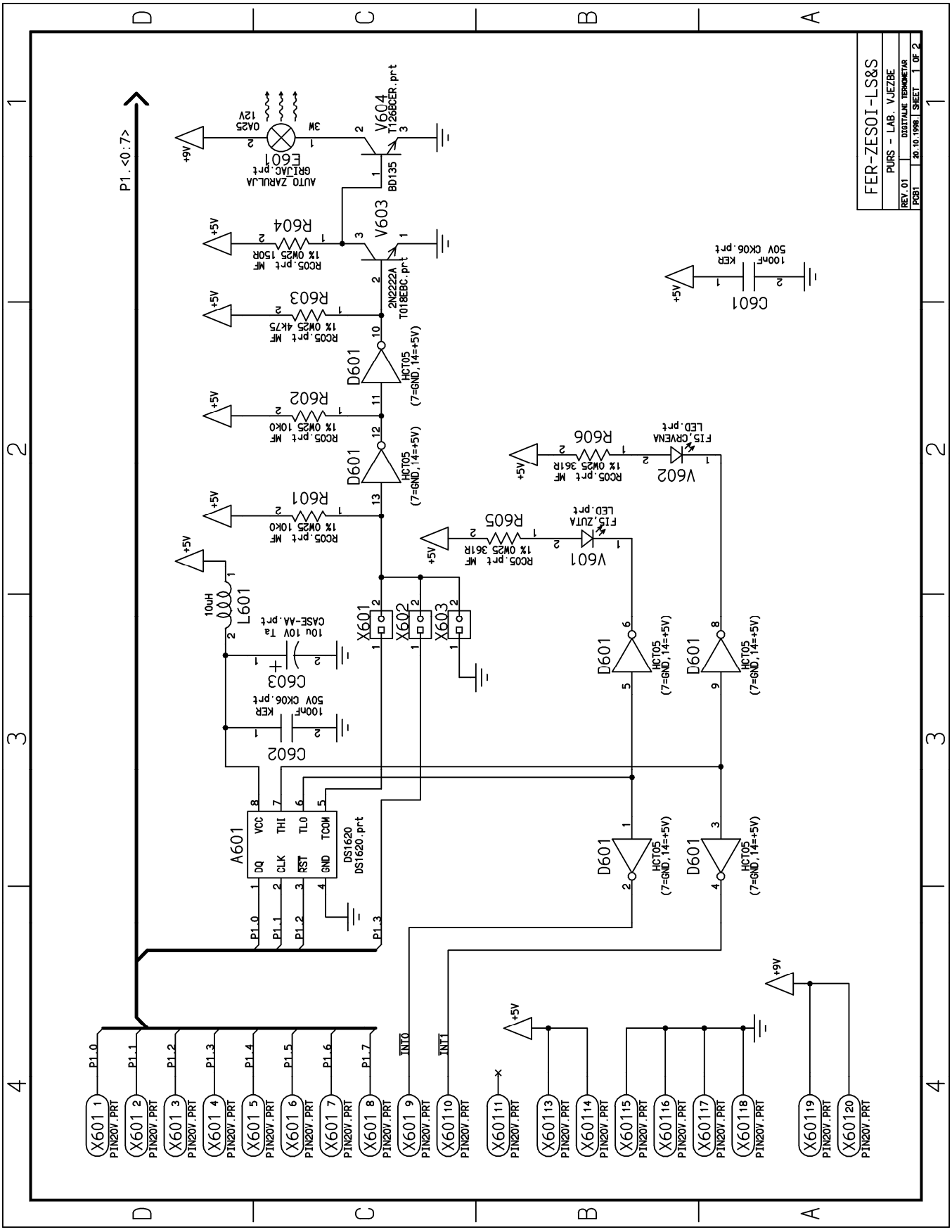
Slika 4.3 Električna shema računalnog sustava s mikrokontrolerom 8031, strana 3/5.



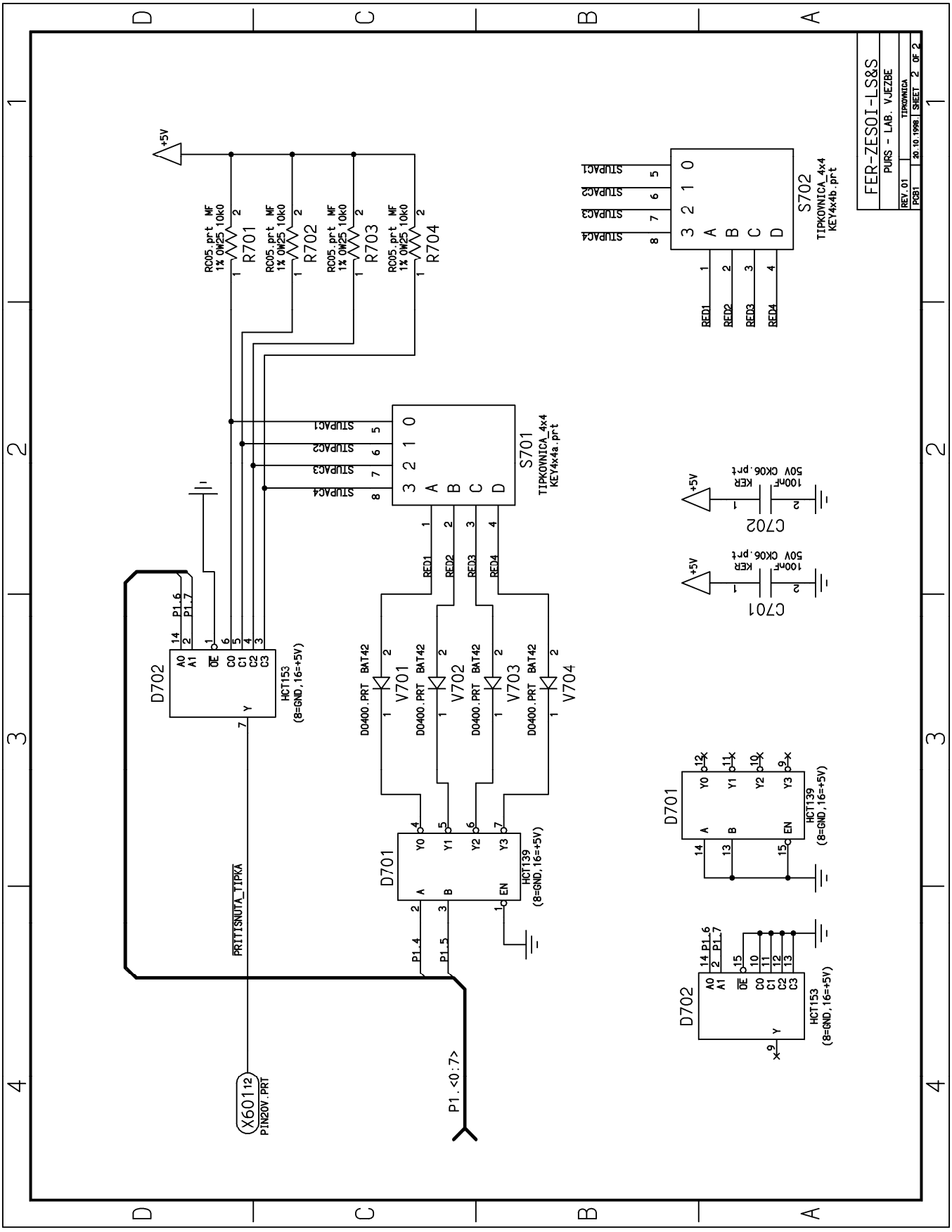
Slika 4.4 Električna shema računalnog sustava s mikrokontrolerom 8031, strana 4/5.



Slika 4.5 Električna shema računalnog sustava s mikrokontrolerom 8031, strana 5/5.



Slika 4.6 Električna shema termometra.



Slika 4.7 Električna shema tipkovnice.

5 Programiranje u asembleru

5.1 Zadatak 1

Potrebno je napisati program u asembleru koji će napraviti provjeru ispravnosti prvih 100 lokacija vanjske podatkovne memorije. Tokom provjere, na svaku memorijsku lokaciju potrebno je upisati broj AAh, pročitati podatak s te lokacije, te provjeriti da li je pročitano AAh. Ponoviti postupak za svih 100 lokacija.

Ako je spomenuti dio memorije ispravan, na digitalnom pokazniku ispisati AA, a ako je neispravan, ispisati 00.

5.2 Upute za rad

Razvoj programske podrške počinje izradom specifikacije programa. Specifikacija obuhvaća ulazne i izlazne parametre, te funkcijske zahtjeve. U našem slučaju napravit ćemo analizu sklopovlja za koje pišemo programsku podršku, te izraditi, odnosno predložiti dijagram toka. Nakon definiranja specifikacije, pristupa se izradi programa koja obuhvaća pisanje koda, prevođenje, povezivanje i punjenje, simulaciju, te ispitivanje na stvarnom sklopovlju.

Obzirom da je ovo korisniku prvi susret s programiranjem mikrokontrolera, a posebno s programskim paketom Keil uVision, navedene faze rada će u daljnjem tekstu biti popraćene detaljnim uputama. Također, radi lakšeg praćenja iznesenog gradiva, njihov prirodan redoslijed bit će djelomično izmijenjen.

5.2.1 Analiza sklopovlja

Pokaznici se "vide" u vanjskom podatkovnom prostoru. Analizom sheme na slikama 4.2 i 4.3 odrediti na kojim adresama se oni nalaze te koji broj je na tu adresu potrebno upisati da bi se na pokazniku pojavio znak 0 ili A.

Digitalni pokaznik V301 (desni) nalazi se na adresi _____ .

Digitalni pokaznik V302 (lijevi) nalazi se na adresi _____ .

Da bi se na pokazniku pojavio znak 0 potrebno je upisati _____ .

Da bi se na pokazniku pojavio znak A potrebno je upisati _____ .

Prva adresa vanjskog RAM-a je _____ .

Posljednja adresa vanjskog RAM-a je _____ .

5.2.2 Prvo pokretanje programa "Keil uVision"

Rad s programskim paketom "Keil uVision" počinje pokretanjem korisničkog sučelja pomoću

Start/Programs/Keil uVision

Nakon pokretanja na ekranu se pojavljuje prozor koji sadrži

- izbornik *(Menu Bar)*: File, Edit, ... Help, koji se nalaze na vrhu prozora
- radni prozor
- jedan ili više pomoćnih prozora za prikaz datoteka projekta, poruka i sl.

Pored ovih elemenata, u prozoru se može vidjeti i

- skupine gumba *(Toolbar)* koje odgovaraju pojedinim alatima, a koji predstavljaju prečice *(Shortcut)* za pojedine komande
- statusna linija *(Status Bar)* na dnu ekrana, itd.

Koji će od ovih elemenata biti vidljivi odabire se pomoću izbornika **View**. Preporučuje se da tokom rada budu uvijek vidljivi prozor za prikaz datoteka projekta *(Project Window)* i prozor za prikaz poruka *(Output Window)*. Ukoliko ovi elementi nisu vidljivi, potrebno ih je aktivirati odabirom

View/Project Window
View/Output Window

Također, potrebno je učiniti vidljivom statusnu liniju *(Status Bar)*, te skup alata za rad s datotekama *(File Toolbar)*, pomoću komandi

View/Status Bar
View/File Toolbar

Preporučuje se isključiti ostale prozore odnosno skupine alata.

5.2.3 Organizacija projekta

Zbog lakšeg snalaženja, u velikom broju programskih paketa definirana je hijerarhija i način označavanja datoteka *(project management)*. U programskom paketu uVision datoteke su grupirane u slijedeće cjeline

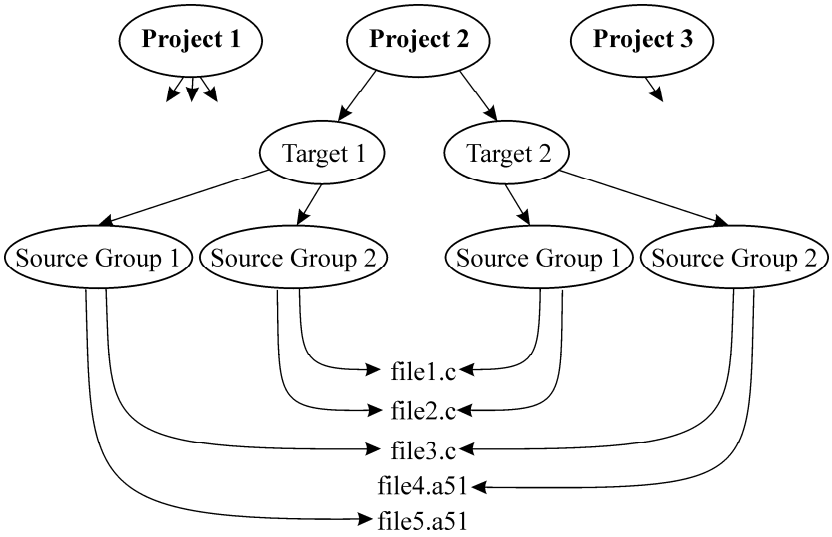
- projekti *(projects)*
- ciljevi *(targets)*
- grupe *(source groups)*

Slika 5.1 prikazuje hijerarhijsku strukturu navedenih cjelina. Sve datoteke koje pripadaju jednom projektu u pravilu se nalaze u jednom direktoriju. Svaki projekt sadrži jedan ili više ciljeva *(target)*. Unutar ciljeva nalaze se datoteke izvornog koda podijeljene u grupe. Pri tome datoteke koje čine srodnu cjelinu pripadaju istoj grupi.

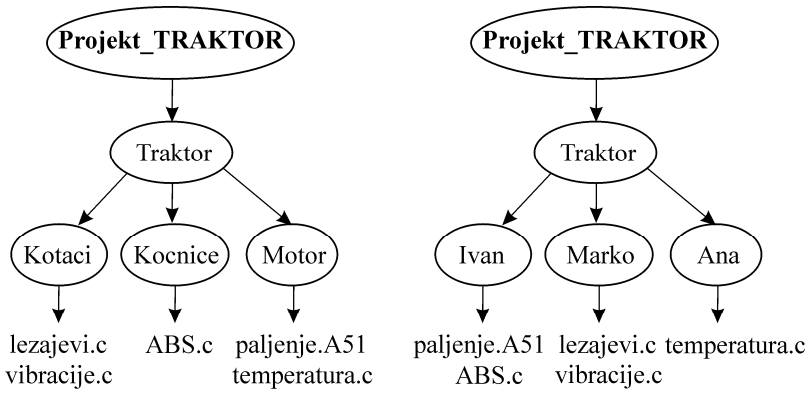
Kao što se vidi sa slike 5.1, iste datoteke izvornog koda pripadaju različitim ciljevima odnosno njihovim grupama. Ovaj pristup omogućava da se istovremeno razvija finalna programska podrška i razvojna programska podrška. Razvojna programska podrška ima ugrađene i neke dijelove koda koji služe uhodavanju, kao što su ispisi poruka i sl. Kad se prevodi *(compile)*, povezuje *(link)* i puni *(load)*, odnosno jednom riječju gradi *(build)* razvojna programska podrška, radi se s jednim ciljem, na primjer "Target 1". Kad je razvoj gotov, finalni program dobiva se prevođenjem, povezivanjem i punjenjem drugog cilja, na primjer "Target 2".

Slika 5.2 prikazuje dva pristupa organizaciji datoteka unutar jednog projekta. U prvom primjeru datoteke s izvornim kodom grupirane su prema funkciji, dok su u drugom slučaju grupirane prema autorima. Drugi slučaj je pogodan kod velikih projekata u kojima na razvoju programske podrške radi velik broj ljudi.

U okviru laboratorijskih vježbi koristit ćemo grupiranje datoteka prema funkciji. Nadalje, koristit ćemo samo jedan cilj, obzirom da je u našem slučaju finalna i razvojna programska podrška identična.



Slika 5.1 Hijerarhijska struktura projekta.



Slika 5.2 Grupiranje datoteka prema funkciji i prema autoru.

5.2.4 Definiranje projekta

Sva rješenja ovog i svih ostalih zadataka bit će pohranjena u korisnikovom direktoriju koji će u daljnjem tekstu biti označavan kao

vlastiti_direktorij

Rješenje ovog zadatka predstavljat će projekt koji ćemo nazvati **proj1**. Potrebno je napraviti direktorij

vlastiti_direktorij\proj1

koji će sadržavati sve datoteke ovog projekta. Pomoću komande

Project/New Project

otvoriti novi projekt čija se konfiguracijska datoteka zove

proj1.Uv2

Datoteku je potrebno spremiti u direktorij

vlastiti_direktorij\proj1

Nakon kreiranja projekta pojavljuje se prozor za odabir mikrokontrolera za koji će se razvijati ciljna programska podrška. Potrebno je odabrati mikrokontroler **80C31**

proizvođača **Philips** ili **NPX**. NE uključivati STARTUP datoteku u program. Uočiti da se je u prozoru koji pokazuje hijerarhiju projekta pojavio cilj "Target 1" s pripadajućom grupom izvornog koda "Source Group 1".

Pomoću komande

File/New

otvoriti novi radni list. Spremiti radni list pomoću komande

File/Save As

kao

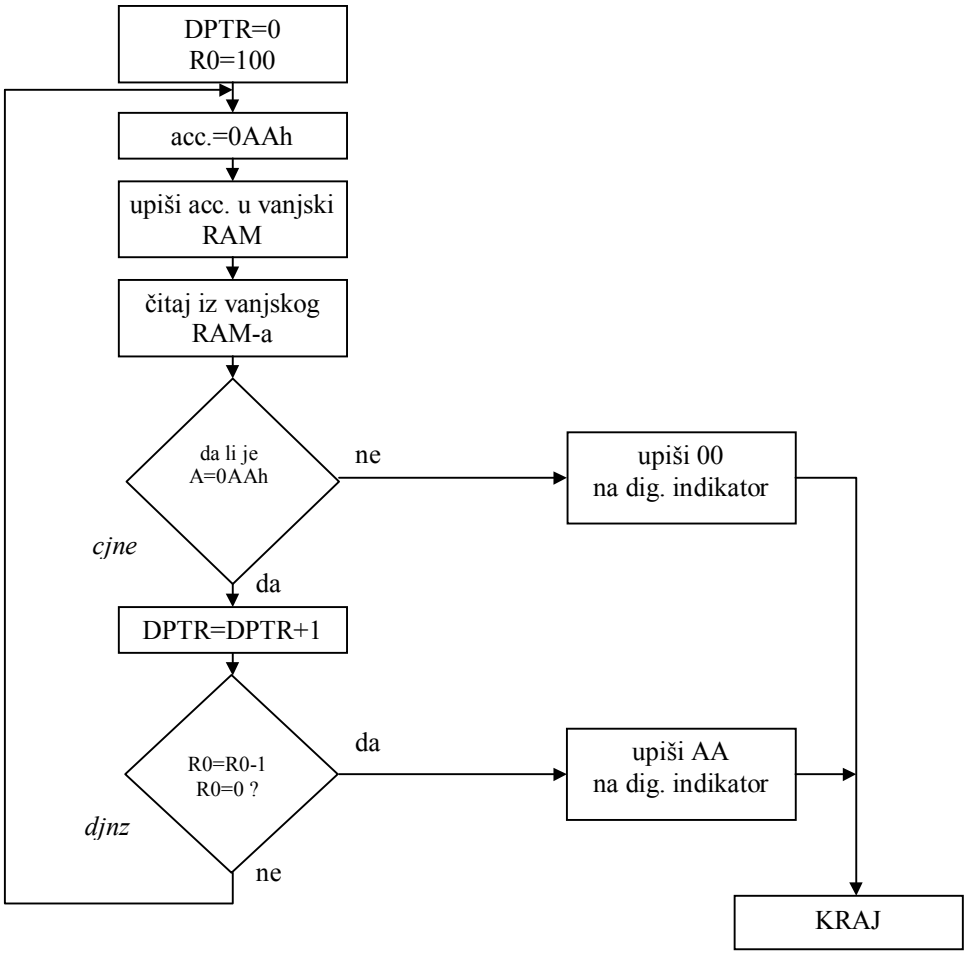
vlastiti_direktorij\zad1.a51

Nastavak **a51** potrebno je OBAVEZNO uključiti u ime datoteke. U daljnjem radu NE ZABORAVITI povremeno, te na kraju rada spremiti datoteku komandom

File/Save

5.2.5 Izrada programa

Na otvoreni radni list sada je potrebno napisati program prema zadanom zadatku. Preporučuje se pisanje programa prema dijagramu toka koji prikazuje slika 5.3. Da bi se olakšao rad korisniku kojem je ovo možda prvi praktični susret s asemblerom mikrokontrolera 8051, na slici 5.3 su navedena imena registara koje je potrebno koristiti, kao i nazivi instrukcija za grananja i petlje.



Slika 5.3 Dijagram toka programa zadatka 1.

5.2.6 Prevođenje

Nakon što je program napisan, njegovu datoteku potrebno je uključiti u grupu **Source Group 1**. To se može učiniti pomoću komande (uVision2)

Project/Targets, Groups, Files...

ili (uVision3)

Project/Manage/Components, Environment, Books...

Da bi prevođenje, povezivanje odnosno punjenje moglo biti ispravno provedeno, potrebno je podesiti opcije. Opcije su vezane uz cilj, tj. u našem slučaju uz "Target 1". Selektirati "Target 1" i pokrenuti komandu

Project/Options for Target 'Target 1'

Prozor koji se pojavio sadrži slijedeće grupe opcija i parametara

- **Device**: odabir konkretnog mikrokontrolera.
- **Target**: definiranje memorijskih modela te specifikacija memorijskih prostora.
- **Output**: definiranje tipa i parametara datoteka koje će nastati prevođenjem, povezivanjem i punjenjem.
- **Listing**: definiranje oblika datoteke koja sadrži ispis postupka prevođenja, ".lst".
- **C51**: definiranje opcija prevodioca jezika C.
- **A51**: definiranje opcija prevodioca koda u asembleru.
- **BL51 Locate**: definiranje opcija programa za punjenje.
- **BL51 Misc**: definiranje opcija programa za punjenje.
- **Debug**: definiranje parametara koji se koriste kod uhodavanja programa u simulatoru.

Selektirati pojedinačno svaku grupu parametara i odabrati predefinirane parametre pritiskom na tipku **Defaults**. Odabrati grupu **Output** te uključiti opciju koja omogućava kreiranje HEX inačice izvršnog programa. Ovako podešene opcije omogućit će nam prevođenje našeg programa kao i generiranje njegove izvršne verzije. U ovoj fazi rada nećemo podrobnije ulaziti u ostale grupe parametara, već će o njima biti više riječi u daljnjem tekstu.

Prevođenje, povezivanje i punjenje pokreće se komandom

Project/Build target

nakon čega se u izlaznom prozoru pojavljuju poruke o uspješnosti ove radnje. Ukoliko ima grešaka, potrebno ih je otkloniti i ponoviti postupak prevođenja. Postupak ponavljati dok se ne dobije poruka **0 Error(s), 0 Warning(s)**. NIJE dopušteno ignorirati poruke tipa "Warning".

5.2.7 Uhodavanje programa u simulatoru

Nakon što je izvršni program uspješno napravljen potrebno je u simulatoru provjeriti njegovu ispravnost. Simulator se pokreće komandom

Debug / Start/Stop Debug Session

nakon čega korisničko sučelje počne prikazivati prozore potrebne za simulaciju rada mikrokontrolera. Pokrenuti komadnu

View

i uočiti koji prozori su na raspolaganju. Uključiti prozore

Project Window koji prikazuje stanje registara mikrokontrolera,
Output Window koji prikazuje poruke,
Memory Window koji pokazuje stanje memorije.

U prozoru koji pokazuje stanje memorije podesiti prikaz vanjske podatkovne memorije počevši od lokacije 0000h. To se postiže upisom **X:0** u polje **Address**. Pritom **X:** označava vanjsku podatkovnu memoriju, a broj koji slijedi početnu adresu od koje se memorija prikazuje. Programskoj memoriji odgovara oznaka **C:**, a unutrašnjoj podatkovnoj memoriji oznaka **D:**.

Resetirati mikrokontroler komandom

Peripherals/Reset CPU

Postupno izvršavati program te provjeriti njegovu ispravnost. Izvršavanje pojedinih naredbi postiže se pokretanjem komandi:

Debug/Step: (ili tipka F11) izvršava jednu instrukciju
Debug/Go ili **Run**: (ili tipka F5) pokreće izvršavanje programa do slijedeće točke na kojoj smo podesili zaustavljanje *(Breakpoint)*. "Breakpoint" se postavlja i briše dvostrukom selekcijom neke instrukcije pomoću miša
Debug/Stop ili **Stop Running**: prekida izvršavanje programa

Nakon što je završeno uhodavanje programa u simulatoru, simulator se isključuje ponavljanjem komande

Debug / Start/Stop Debug Session

5.2.8 Uhodavanje programa na sklopovlju

Nakon što je program uhodan u simulatoru, može se početi s uhodavanjem na sklopovlju. Prije toga potrebno je datoteku proj1.hex (ako nije nazvana nekim dugim imenom) koja sadrži izvršni kod u Intel-HEX formatu prevesti u binarni format. To se izvodi pomoću programa **hexbin.exe** koja se nalazi na web stranici predmeta **Ugradbeni računalni sustavi**, podstranici **Laboratorij** u repozitoriju **2. vježba - repozitorij datoteka**. Datoteku je potrebno presnimit u

vlastiti_direktorij\proj1

Pokrenuti komandni prozor, prijeći u **vlastiti_direktorij\proj1** i pokrenuti komandu

hexbin proj1.hex proj1.bin i ff

Postaviti EPROM-emulator u stanje **load** te učitati podatke u emulator kopiranjem datoteke na paralelno sučelje osobnog računala pomoću

copy proj1.bin lpt1: /b

Prebaciti sklopku na emulatoru u položaj **emulator**, resetirati mikrokontroler i provjeriti ispravnost rada programa. U slučaju potrebe otkloniti greške.

5.2.9 Rješenje zadatka

Nakon što je program napisan i uhodan, potrebno ga je (rukom) prepisati u dani prostor.

6 Deklaracija registara posebne namjene

6.1 Zadatak 2

Proučiti sadržaj datoteke reg51.h . Uočiti zašto ovakva datoteka mora biti uključena u praktički svaki modul koji sadrži C kod za mikrokontroler familije 8051.

6.2 Upute za rad

Pronaći mjesto na disku na koje je instaliran programski paket Keil - uVision. Prekopirati datoteku

.\Keil\c51\inc\reg51.h

u direktorij

vlastiti_direktorij

Pomoću editora teksta **notepad** pregledati njen sadržaj, te opisati što se u njoj nalazi.

_____

_____

_____

_____

_____

Uočiti da registar P1 nema deklarirane varijable za pojedine bitove. Napisati za primjer bitova 0 i 1 kako bi trebala izgledati spomenuta deklaracija?

_____

_____

_____

_____

_____

U kojem dijelu programa bi to trebalo napisati, ako deklaracija nije dio datoteke reg51.h?

_____

_____

_____

_____

_____

7 Serijska veza

7.1 Zadatak 3

Potrebno je napisati program koji preko serijske veze šalje na terminal poruku "Zadatak 3 uspjesno napravljen". Brzinu prijenosa serijske veze treba podesit na 2400 bitova/s.

7.2 Upute za rad

Rezultat ovog i slijedećih zadataka bit će programska podrška sustava za mjerenje i regulaciju temperature. Iako svaki zadatak čini tematsku cjelinu, svaki novi program bit će integriran u cjelokupnu programsku podršku. U praksi to znači da ovaj i slijedeći zadaci čine jedan projekt, u koji ćemo dodavati novostvorene datoteke, te ih uhodavati jednu po jednu. Ovakva organizacija odabrana je da bi se što vjernije ilustrirao stvaran razvoj programske podrške.

Postupak razvoja zadanog programa podijelit ćemo u slijedeće faze

- definiranje projekta
- izrada programa
- prevođenje, povezivanje i punjenje
- uhodavanje programa na simulatoru
- uhodavanje programa na sklopovlju

U sklopu pojedinih faza bit će dane teorijske napomene potrebne za rad, te opisan sam postupak rada.

7.2.1 Definiranje projekta

Programska podrška razvijana u okviru vježbe predstavljat će projekt koji ćemo nazvati **proj2**. Potrebno je napraviti direktorij

vlastiti_direktorij\proj2

koji će sadržavati sve datoteke ovog projekta.

Pokrenuti programski paket "Keil uVision" pomoću

Start/Programs/Keil uVision

Pomoću komande

Project/New Project

otvoriti novi projekt čija se konfiguracijska datoteka zove

proj2.Uv2

Datoteku je potrebno spremiti u direktorij

vlastiti_direktorij\proj2

Nakon kreiranja projekta pojavljuje se prozor za odabir mikrokontrolera za koji će se razvijati ciljna programska podrška. Kao i u zadatku 1 potrebno je odabrati mikrokontroler **80C31** proizvođača **Philips** ili **NPX**. NE uključivati STARTUP datoteku u program.

Pomoću komande

File/New

otvoriti novi radni list. Spremiti radni list pomoću komande

File/Save As

kao

vlastiti_direktorij\proj2\glavni.c

Nastavak **c** potrebno je OBAVEZNO uključiti u ime datoteke.

Pokrenuti komandu (uVision2)

Project/Targets, Groups, Files...

ili (uVision3)

Project/Manage/Components, Environment, Books...

te promijeniti ime "Source Group 1" u "Zadatak 3". Dodati datoteku glavni.c u grupu "Zadatak 3".

U daljnjem radu NE ZABORAVITI povremeno, te na kraju rada spremiti datoteku komandom

File/Save

7.2.2 Izrada programa

Najjednostavniji način slanja poruka preko serijske veze je slanje pomoću funkcije **printf**. Ulazno izlazne funkcije kao što su printf, scanf i druge prevedene pomoću prevodioca Keil C51 koristit će serijsko sučelje kao ulaz ili izlaz. Prototipovi ovih funkcija nalaze se u datoteci stdio.h pa program mora sadržavati liniju

#include <stdio.h>

Da bi komunikacija preko serijskog sučelja mogla biti uspostavljena, potrebno je inicijalizirati sučelje na odgovarajući način. Inicijalizacija uključuje

- inicijalizaciju brojila T1 koji određuje brzinu prijenosa serijske veze

```
TMOD=0x20;  /* Timer1:Gate|C/T|M1|M0| Timer0:Gate|C/T|M1|M0|
               /* Brojilo T1 radi na način rada 2. */
TH1=0xF4;   /* Broj za ponovno punjenje brojila T1 => 2400 bitova/s. */
TL1=0xF4;   /* Početni sadržaj brojila T1          => 2400 bitova/s. */
TR1=1;      /* TF1|TR1|TF0|TR0|IE1|IT1|IE0|IT0 */
               /* Bit u TCON registru koji pokreće rad brojila. */
```

- inicijalizaciju serijskog sučelja

```
SCON=0x52;  /* SM0|SM1|SM2|REN|TB8|RB8|TI|RI */
               /* Serijsko sučelje radi na način rada 2. */
```

Napisati zadani program. Kao podsjetnik na strukturu C programa po potrebi se poslužiti primjerom iz poglavlja 3.1.4.

Uočiti za se ovaj program izvršava samo jednom. Zato na kraju mora sadržavati beskonačnu petlju.

7.2.3 Prevođenje, povezivanje i punjenje

Da bi prevođenje, povezivanje odnosno punjenje moglo biti ispravno provedeno, potrebno je podesiti opcije. Opcije su vezane uz cilj, tj. u našem slučaju uz "Target 1". Selektirati "Target 1" i pokrenuti komandu

Project/Options for Target 'Target 1'

U prozoru koji se pojavio potrebno je podesiti slijedeće grupe opcija i parametara

- **Device**: odabir konkretnog mikrokontrolera.
  - Provjeriti je li podešen odgovarajući tip mikrokontrolera.
- **Target**: definiranje memorijskih modela te specifikacija memorijskih prostora.
  - Podesiti memorijski model na **small**.
  - Podesiti frekvenciju takta na iznos 11.0592 MHz.
  - Podesiti "Off-chip Code memory" i "Off-chip Xdata memory" na odgovarajuće vrijednosti. Poslužiti se shemama sklopa danim u poglavlju 4.
- **Output**: definiranje tipa i parametara datoteka koje će nastati prevođenjem, povezivanjem i punjenjem.
  - Uključiti opciju za kreiranje izlazne datoteke u HEX formatu.

Preostale parametre postaviti u njihove predefinirane vrijednosti, pomoću tipke **Default**.

- **Listing**: definiranje oblika datoteke koja sadrži ispis postupka prevođenja, ".lst".
- **C51**: definiranje opcija prevodioca jezika C.
- **A51**: definiranje opcija prevodioca koda u asembleru.
- **BL51 Locate**: definiranje opcija programa za punjenje.
- **BL51 Misc**: definiranje opcija programa za punjenje.
- **Debug**: definiranje parametara koji se koriste kod uhodavanja programa u simulatoru.

Prevođenje, povezivanje i punjenje pokreće se komandom

Project/Build target

nakon čega se u izlaznom prozoru pojavljuju poruke o uspješnosti ove radnje. Ukoliko ima grešaka, potrebno ih je otkloniti i ponoviti postupak prevođenja. Postupak ponavljati dok se ne dobije poruka **0 Error(s), 0 Warning(s)**. NIJE dopušteno ignorirati poruke tipa "Warning".

7.2.4 Uhodavanje programa u simulatoru

Nakon što je izvršni program uspješno napravljen potrebno je u simulatoru provjeriti njegovu ispravnost. Simulator se pokreće komandom

Debug / Start/Stop Debug Session

nakon čega korisničko sučelje počne prikazivati prozore potrebne za simulaciju rada mikrokontrolera. Pokrenuti komadnu

View

i uključiti prozore

Project Window koji sad prikazuje stanje registara mikrokontrolera,
Output Window koji prikazuje poruke,
Memory Window koji pokazuje stanje memorije.
Serial Window #0 koji pokazuje poruke poslane preko serijske veze.

Resetirati mikrokontroler komandom

Peripherals/Reset CPU

Postupno izvršavati program te provjeriti njegovu ispravnost. Izvršavanje pojedinih naredbi postiže se pokretanjem komandi:

Debug/Step: (ili tipka F11) izvršava jednu instrukciju
Debug/Go ili **Run**: (ili tipka F5) pokreće izvršavanje programa do slijedeće točke na kojoj smo podesili zaustavljanje *(Breakpoint)*. "Breakpoint" se postavlja i briše dvostrukom selekcijom neke instrukcije pomoću miša
Debug/Stop: (ili tipka ESC) prekida izvršavanje programa

Nakon što je završeno uhodavanje programa u simulatoru, simulator se isključuje ponavljanjem komande

Debug / Start/Stop Debug Session

Provjeriti ispravnost programa u simulatoru. Pri tome se u prozoru koji prikazuje stanje serijske veze mora ispisati tražena poruka "Zadatak 3 uspjesno napravljen".

7.2.5 Uhodavanje programa na sklopovlju

Nakon što je program uhodan u simulatoru, može se početi s uhodavanjem na sklopovlju. Prije toga potrebno je datoteku proj2.hex (ako nije nazvana nekim dugim imenom) koja sadrži izvršni kod u Intel-HEX formatu prevesti u binarni format. To se izvodi pomoću programa **hexbin.exe** koja se nalazi na web stranici predmeta **Ugradbeni računalni sustavi**, podstranici **Laboratorij** u repozitoriju **2. vježba - repozitorij datoteka**. Datoteku je potrebno presnimit u

vlastiti_direktorij\proj2

Pokrenuti komandni prozor, prijeći u **vlastiti_direktorij\proj2** i pokrenuti komandu

hexbin proj2.hex proj2.bin i ff

Postaviti EPROM-emulator u stanje **load** te učitati podatke u emulator kopiranjem datoteke na paralelno sučelje osobnog računala pomoću

copy proj2.bin lpt1: /b

Prebaciti sklopku na emulatoru u položaj **emulator**, resetirati mikrokontroler i provjeriti ispravnost rada programa. U slučaju potrebe otkloniti greške.

Nakon reseta, mikrokontroler će poslati preko serijske veze poruku zadanu poruku. Da bismo tu poruku mogli prihvatiti i prikazati potreban nam je terminal. Kao što je spomenuto u uvodnim poglavljima ove vježbe, kao terminal ćemo upotrijebiti osobno računalo s odgovarajućom programskom podrškom. Pokrenuti program

Start/Programs/Accessories/Communications/Hyper Terminal

Pomoću komande

File/New Connection

otvoriti vezu pod nazivom npr. "SerijskaVeza". Podesiti vezu terminala i našeg računalnog sustava *(Connection using)* tako da koristi serijsko sučelje osobnog računala na koji je spojena maketa. Podesiti parametre veze kako je to dano u zadatku

- 2400 bitova/s
- 8 bitova podataka
- prijenos bez pariteta
- 1 stop bit
- isključeno upravljanje prijenosom *(Flow control =>None)*

Spajanje odnosno odspajanje veze izvodi se komandama Call/Call i Call/Disconnect.

Resetirati mikrokontroler i provjeriti ispravnost rada programa.

7.2.6 Rješenje zadatka

Nakon što je program napisan i uhodan u simulatoru, potrebno ga je (rukom) prepisati u dani prostor.

8 Provjera ispravnosti memorije

8.1 Zadatak 4

U program iz prethodnog zadatka dodati odsječak koji provjerava ispravnost cijele vanjske podatkovne memorije. Provjeru raditi pomoću upisa i čitanja broja 0xAA. Poruku "Memorija je ispravna." ili "Memorija nije ispravna." ispisati preko serijske veze.

8.2 Upute za rad

8.2.1 Analiza sklopovlja

Analizom sheme na slici 4.2 odrediti:

Prva adresa vanjskog RAM-a je _____ .

Posljednja adresa vanjskog RAM-a je _____ .

8.2.2 Izrada programa

Provjera ispravnosti memorije radi se tako da se na lokaciju upiše 0xAA i nakon toga pročita te provjeri da li je pročitan ispravan broj. Pisanje i čitanje iz vanjske memorije s točno određene lokacije mora se raditi preko pokazivača, kao što je pokazano u prethodnom zadatku. Pisanje i čitanje imaju oblik:

```
*mem=0xAA;
vtest=*mem;
```

gdje je **mem** pokazivač na vanjsku memoriju. Takav pokazivač u pravilu treba biti deklariran kao **volatile**.

Za prolaz kroz cijelu memoriju koristiti "for" petlju. Pritom se preporučuje u petlji postaviti zastavicu koja pokazuje se li memorija ispravna. Nakon završetka petlje treba, ovisno o stanju zastavice, preko serijske veze poslati poruku "Memorija je ispravna." ili "Memorija nije ispravna".

8.2.3 Uhodavanje programa

Za uhodavanje programa koristiti simulator i sklopovlje.

Pokretanje programa na sklopovlju mora rezultirati porukom "Memorija je ispravna.". Nakon toga je potrebno promijeniti program postavljanjem gornje granice memorijskog prostora u područje u kojem nema fizičke memorije. Ponovo pokrenuti prevođenje, punjenje itd. Ako se ovakav program pokrene na sklopovlju, mora rezultirati porukom "Memorija nije ispravna.".

9 Korisničko sučelje

9.1 Zadatak 5

Napisati funkciju koja ispisuje broj na digitalni pokaznik. Funkcija treba imati prototip oblika

void ispis(unsigned char, unsigned char);

pri čemu je prvi argument funkcije karakter 'l' ili 'd', a on određuje na koji pokaznik se upisuje (lijevi ili desni). Drugi argument je broj od 0 do 9 koji se prikazuje.

U glavni program dodati pozive funkcije koji ispisuju na pokazniku broj 45.

9.2 Upute za rad

Kao što je pokazano u zadatku 1, pokaznici se vide u vanjskom podatkovnom prostoru na adresama 0x8000 (lijevi) i 0x8001 (desni). Za ispis pojedinih znamenki potrebno je upisati slijedeće brojeve:

Znak na pokazniku	Broj koji treba upisati
0	0x77
1	0x14
2	0xB3
3	0xB6
4	0xD4
5	0xE6
6	0xE7
7	0x34
8	0xF7
9	0xF6

Upis u vanjsku memoriju na točno određenu lokaciju mora se raditi preko pokazivača. Potrebno je deklarirati pokazivač

unsigned char xdata *pok;

Na primjer, upis karaktera 0x0A na lokaciju 0x8000 radi se na slijedeći način

```
pok=0x8000;
*pok=0x0A;
```

Za upis pojedinih znamenki koristiti "switch/case" ili "if" naredbu. Funkciju treba napisati u datoteku glavni.c koja je napravljena u prethodnom zadatku, te dodati potrebne pozive funkcije koji ispisuju na pokazniku broj 45.

Za uhodavanje programa koristiti simulator i sklopovlje.

10 Serijske vanjske jedinice

10.1 Zadatak 6

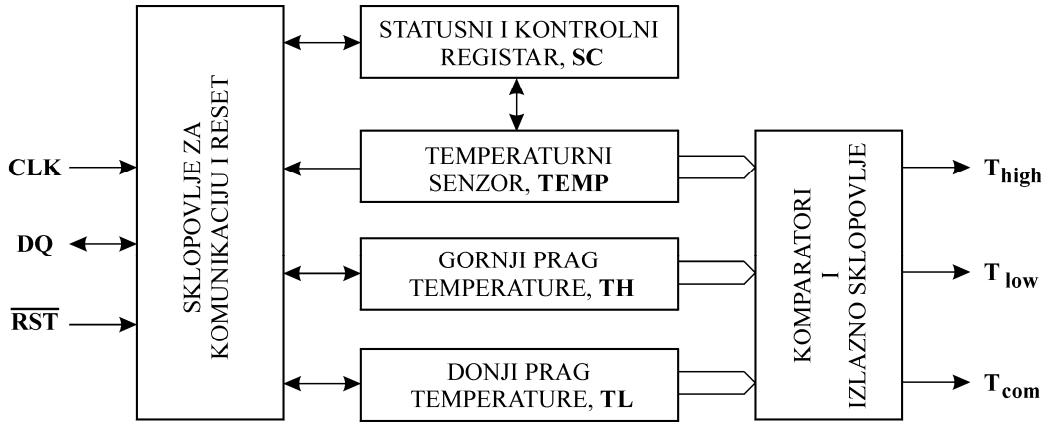
Na ulazno-izlazni konektor računalnog sustava spojeno je sklopovlje termometra. Korištenjem danih funkcija za rad s termometrom potrebno je napisati programsku podršku za mjerenje temperature. Temperaturu iz termometra treba neprekidno očitavati, te preko serijske veze na terminal slati poruku o temperaturi u obliku "Izmjerena je temperatura od XX stupnjeva".

10.2 Upute za rad

10.2.1 Integrirani digitalni termometar DS1620

Za mjerenje temperature koristit će se programabilni termometar/termostat DS1620. Ovaj termometar može mjeriti temperaturu od −55°C do +125°C s razlučivošću 0.5°C. Točnost mjerenja u području od 0°C +70°C iznosi ±0.5°C. Spomenuti sklop nalazi se na pločici koja je opisana u poglavlju 2.

Slika 10.1 prikazuje blokovsku shemu integriranog kruga DS1620. Spomenuti integrirani krug je digitalni termometar i termostat koji može raditi u kombinaciji s računalnim sustavom ili samostalno. Spoj s računalnim sustavom izvodi se pomoću tri signalne linije, CLK, DQ i $\overline{\text{RST}}$. Podaci se prenose preko serijske veze sinkrono sa signalom takta, pa govorimo o sinkronom serijskom prijenosu.



Slika 10.1 Blokovska shema integriranog kruga DS1620.

Termometar sadrži jedan 8-bitni statusni i upravljački registar, **SC**, 9-bitni registar u kojem je pohranjena izmjerena temperatura, **TEMP**, te dva 9-bitna registra koji sadrže gornju i donju temperaturu, **TH** i **TL**, u slučaju kad termometar radi kao termostat. Iz registra koji sadrži temperaturu može se samo čitati, dok se u ostale registre može pisati i čitati.

10.2.2 Funkcije za rad s termometrom

Detaljan opis termometra čitatelj može naći u [9]. Nadalje, u [8] je detaljno obrađena komunikacija između termometra i računalnog sustava, te su dane smjernice za izradu funkcija za komunikaciju s termometrom. Zbog kratkoće vremena, na ovoj laboratorijskoj vježbi bit će korištene tri gotove funkcije oblika

void PisiControl (unsigned char)
void PisiKomandu (unsigned char)
int CitajTemp (void)

Ove funkcije nalaze se u datoteci **termo.c** koja se može naći na web stranici predmeta.

Na početku rada s termometrom potrebno ga je inicijalizirati, tj. u njegov registar SC upisati odgovarajući podatak. Registar SC sadrži 8 bitova

DONE	THF	TLF	NVB	1	0	CPU	1SHOT

čije je značenje

DONE 0=mjerenje temperature u toku
1=mjerenje temperature završeno

THF kad temperatura naraste iznad TH, postavlja se u 1
(ne briše se automatski već ga treba izbrisati upisivanjem 0)

TLF kad temperatura padne ispod TL, postavlja se u 1
(ne briše se automatski već ga treba izbrisati upisivanjem 0)

NVB 1=u toku je upis u EEPROM (10ms do 50ms)
0=nije u toku upis u EEPROM

CPU 1=DQ, CLK i $\overline{\text{RST}}$ koriste se isključivo za komunikaciju
0=kad je $\overline{\text{RST}}$ nisko, 0 na CLK priključku pokreće mjerenje temp.
(koristi se kad DS1620 samostalno radi kao termostat)

1SHOT 1=nakon komande za početak mjerenja, napravi se jedno mjerenje
0=termometar neprekidno mjeri temperaturu

U našem slučaju termometar treba neprekidno mjeriti temperaturu, a ona će se čitati preko linija DQ, CLK i $\overline{\text{RST}}$. Zato sadržaj registra SC mora biti 0x02. Ovaj podatak se u registar upisuje pomoću funkcije

void PisiControl (unsigned char)

Nakon što je termometar konfiguriran moguće je pokrenuti ili zaustaviti mjerenje temperature. To se izvodi pomoću funkcije

void PisiKomandu (unsigned char)

Pritom se mjerenje pokreće komandom 0xEE, a zaustavlja komandom 0x22. U našem slučaju potrebno je pokrenuti mjerenje i više ga ne zaustavljati.

Za čitanje podatka o temperaturi koristi se funkcija

int CitajTemp (void)

Ova funkcija vraća sadržaj registra TEMP. Obzirom da je razlučivost termometra 0.5°C vraćeni podatak predstavlja DVOSTRUKU vrijednost temperature.

10.2.3 Definiranje radne okoline

S web stranice predmeta **Ugradbeni računalni sustavi**, podstranici **Laboratorij** u repozitoriju **2. vježba - repozitorij datoteka** presnimiti datoteku

termo.c

u direktorij

vlastiti_direktorij\proj2

Pokrenuti programski paket "Keil uVision" pomoću

Start/Programs/Keil uVision

Pomoću komande

Project/Open Project

otvoriti projekt čija se konfiguracijska datoteka zove

vlastiti_direktorij\proj2\proj2.Uv2

Pokrenuti komandu (uVision2)

Project/Targets, Groups, Files...

ili (uVision3)

Project/Manage/Components, Environment, Books...

te dodati novu grupu pod nazivom "Zadatak 6" i u nju dodati datoteku **termo.c**.

10.2.4 Analiza sklopovlja

Proučiti električnu shemu danu u prilogu i odgovoriti na koje priključke mikrokontrolera su spojene linije za komunikaciju s termometrom DS1620?

_____

_____

10.2.5 Analiza funkcija za komunikaciju s termometrom

Proučiti datoteku **termo.c** i odgovoriti na koji način su priključci is prethodnog pitanja učinjeni vidljivim u programskom okruženju?

_____

_____

10.2.6 Mjerenje i ispis temperature

U postojeću, tj. dosad napravljenu programsku podršku, potrebno je dodati dio za inicijalizaciju termometra i pokretanje temperature, te programski odsječak u kojem se temperatura neprekidno mjeri, te ispisuje na terminalu.

Ne zaboraviti da podatak koji vraća funkcija **CitajTemp** predstavlja dvostruku vrijednost temperature. Obzirom da školska inačica programskog paketa Keil uVision ne podržava aritmetiku pokretnog zareza, potrebno je koristiti cjelobrojno dijeljenje.

Dobivenu temperaturu potrebno je preko serijske veze poslati na terminal korištenjem funkcije **printf**.

11 Regulacija temperature

11.1 Zadatak 7

U napravljenu programsku podršku potrebno je dodati odsječak za dvopoložajnu regulaciju temperature. Kad temperatura padne ispod 30°C, potrebno je uključiti grijalicu (žarulju). Kad temperatura poraste iznad 32°C, potrebno je isključiti grijalicu.

11.2 Upute za rad

11.2.1 Analiza sklopovlja

Proučiti električnu shemu danu u prilogu i odgovoriti na koji priključak mikrokontrolera je spojena grijalica (žarulja)?

_____

_____

_____

Uočiti da se grijalica uključuje upisom nule, a isključuje upisom jedinice.

11.2.2 Regulacija temperature

Za pristup grijalici potrebno je deklarirati varijablu **grijalica** koja predstavlja bit u SFR prostoru, korištenjem deklaracije tipa **sbit**.

Za regulaciju temperature koristiti ćemo mikrokontroler, a ne funkcije termostata koje postoje kod integriranog kruga DS1620. Regulacija se najlakše izvodi pomoću dvije **if** naredbe.

12 Analiza reentrant funkcije

12.1 Zadatak 8

Napisati zadane funkcije, prevesti kod i analizirati dobiveni rezultat.

```
char funk1 (char a)
   {
   char b=5;
   b=a+b;
   return b;
   }

char funk2 (char a) using 1
   {
   char b=5;
   b=a+b;
   return b;
   }

char funk3 (char a) reentrant
   {
   char b=5;
   b=a+b;
   return b;
   }
```

12.2 Upute za rad

12.2.1 Definiranje radne okoline

Da ne bismo unosili neželjen kod u projekt proj2 na kojem ćemo još raditi kasnije, za potrebe ovog zadatka otvorit ćemo novi projekt pod nazivom proj3. Napraviti direktorij

vlastiti_direktorij\proj3

i pokrenuti programski paket "Keil uVision". Pomoću komande

Project/New Project

Zatvoriti otvorene projekte, te otvoriti novi projekt čija se konfiguracijska datoteka zove

proj3.Uv2

a nalazi se u direktoriju

vlastiti_direktorij\proj3

Nakon kreiranja projekta pojavljuje se prozor za odabir mikrokontrolera za koji će se razvijati ciljna programska podrška. Kao i u zadatku 1 potrebno je odabrati mikrokontroler **80C31** proizvođača **Philips** ili **NPX**. NE uključivati STARTUP datoteku u program.

Pomoću komande

File/New

otvoriti novi radni list i spremiti ga kao

vlastiti_direktorij\proj3\zad8.c

Pomoću komande (uVision2)

Project/Targets, Groups, Files...

ili (uVision3)

Project/Manage/Components, Environment, Books...

te promijeniti ime "Source Group 1" u "Analiza". Dodati datoteku **zad8.c** u grupu "Analiza".

12.2.2 Analiza funkcija

Napisati kod zadanih funkcija. Pomoću

Project/Options for Target 'Target 1'

podesiti opcije vezane uz "Target 1". Sve opcije treba postaviti u njihove predefinirane, tj. **Default** vrijednosti. Pokrenuti prevođenje. Ukoliko u kodu nema grešaka, pojaviti će se poruke tipa "Warning" koja govore da funkcije nisu niotkuda pozvane. Te poruke treba zanemariti.

Pokrenuti simulator pomoću

Debug / Start/Stop Debug Session

Pomoću

View/Disassembly Window

uključiti prozor koji sadrži prikaz koda u jeziku C zajedno s pripadajućim asemblerskim odsječcima. Pronaći na kojim adresama u programskoj memoriji počinju naše funkcije.

Funkcija **funk1** počinje na adresi _____

Funkcija **funk2** počinje na adresi _____

Funkcija **funk3** počinje na adresi _____

Na odgovarajućim adresama pronaći kod naših funkcija i prepisati ga.

funk1: funk2: funk3:

U čemu se funkcije razlikuju? Koje karakteristične dijelove sadrže pojedine funkcije?

funk1:

funk2:

funk3:

Kako se u funkcije prenose argumenti? Kako se vraća izlazna vrijednost?

funk1:

funk2:

funk3:

Gdje su pohranjene lokalne varijable ?

funk1:

funk2:

funk3:

Koja je uloga pokazivača ?C_IBP ? Na kojoj memorijskoj lokaciji se on nalazi?

13 Analiza prekidne funkcije

13.1 Zadatak 9

Napisati zadanu funkciju, prevesti kod i analizirati dobiveni rezultat.

```
unsigned char c,d;

void int_funk (void) interrupt 0
   {
   c=d+1;
   }
```

13.2 Upute za rad

13.2.1 Definiranje radne okoline

Otvoriti novi radni list i spremiti ga kao

vlastiti_direktorij\proj3\zad9.c

te dodati datoteku **zad9.c** u grupu "Analiza".

13.2.2 Analiza funkcije

Napisati kod zadane funkcije. Pokrenuti prevođenje. Nakon što je prevođenje uspješno provedeno, proučiti poruke sustava za povezivanje i punjenje, te odgovoriti zašto nema poruke tipa "Warning" koja bi govorila da ova funkcija nije niotkuda pozvana?

Pokrenuti simulator i prepisati kod u asembleru koji odgovara zadanim funkcijama.

Sadržaj kojih registara se sprema na početku programa? Zašto? Zašto se ne sprema sadržaj niti jednog drugog registra?

14 Pisanje prekidnih funkcija

14.1 Zadatak 10

U okviru prethodnih zadataka, napisan je program koji na ekranu terminala (osobnog računala) neprekidno ispisuje podatak o temperaturi, dok istovremeno radi regulaciju temperature u granicama od 30°C do 32°C.

U postojeći program potrebno je dodati prekidnu funkciju koja na digitalnom pokazniku broji sekunde od 0 do 99 i opet počinje od 0. Istovremeno, na ekranu terminala treba i dalje neprekidno ispisivati podatak o temperaturi i raditi regulaciju.

14.2 Upute za rad

14.2.1 Definiranje radne okoline

Sve funkcije razvijene na ovoj vježbi bit će smještene u novu datoteku, te pridružene novoj grupi *(Source Group)* pod nazivom "Zadatak 10". Pokrenuti program Keil - uVision, te pomoću komande

Project/Open Project

otvoriti projekt čija se konfiguracijska datoteka zove

vlastiti_direktorij\proj2\proj2.Uv2

Pomoću komande

File/New

otvoriti novi radni list. Spremiti radni list kao

vlastiti_direktorij\proj2\sat.c

Dodati novu grupu koja se zove "Zadatak 10" i uključiti datoteku **sat.c** u tu grupu.

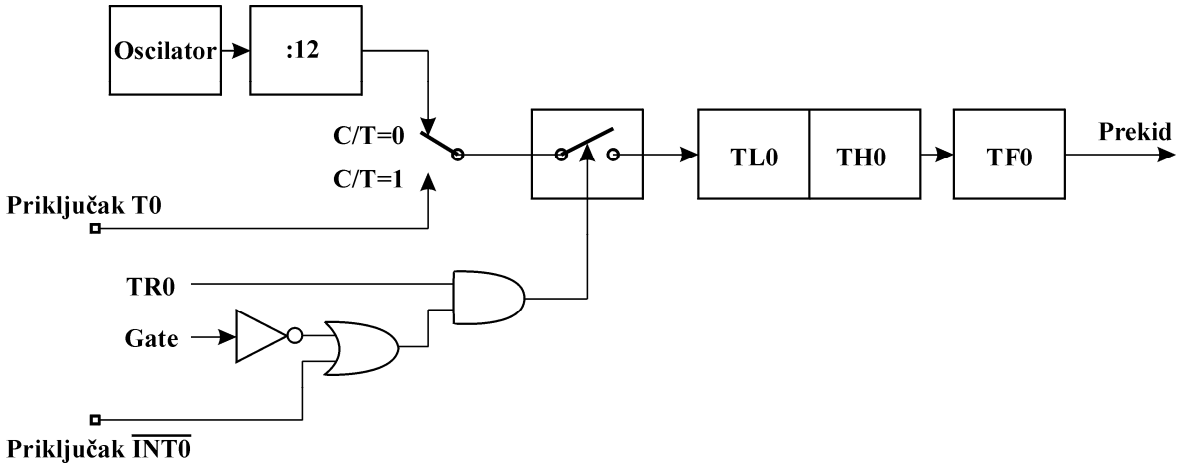
14.2.2 Izrada prekidne funkcije

Brojenje treba izvesti u prekidnoj funkciji, pomoću prekida koji daje brojilo T0. Predvidjeti niži prioritet ovog prekida. Preporučuje se koristiti brojilo koje radi na način 1. Obzirom da brojilo može izbrojiti najviše 2^{16} impulsa, tj. strojnih ciklusa, to neće biti dovoljno za odbrojavanje vremena od 1s. Zato je potrebno u prekidnoj funkciji brojati broj ulazaka i kad se postigne 1 sekunda, promijeniti broj na pokazniku.

Podatak o proteklom broju sekundi potrebno je rastaviti na dvije znamenke, desetice i jedinice. Desetice je najjednostavnije dobiti pomoću cjelobrojnog dijeljenja s 10. Jedinice se mogu dobiti kao ostatak kod cjelobrojnog dijeljenja, za što se koristi operator %. Dobivene desetice i jedinice potrebno je ispisati na pokaznike pomoću funkcije **ispis** koja je razvijena u sklopu zadatka 5.

Paralelno prekidnoj funkciji glavni program neka neprekidno na terminalu ispisuje podatak o temperaturi i radi regulaciju temperature. Obzirom da se digitalni pokaznik sada koristi za ispis sekundi, potrebno je ukloniti pozive funkcije za ispis broja 45 iz glavnog programa.

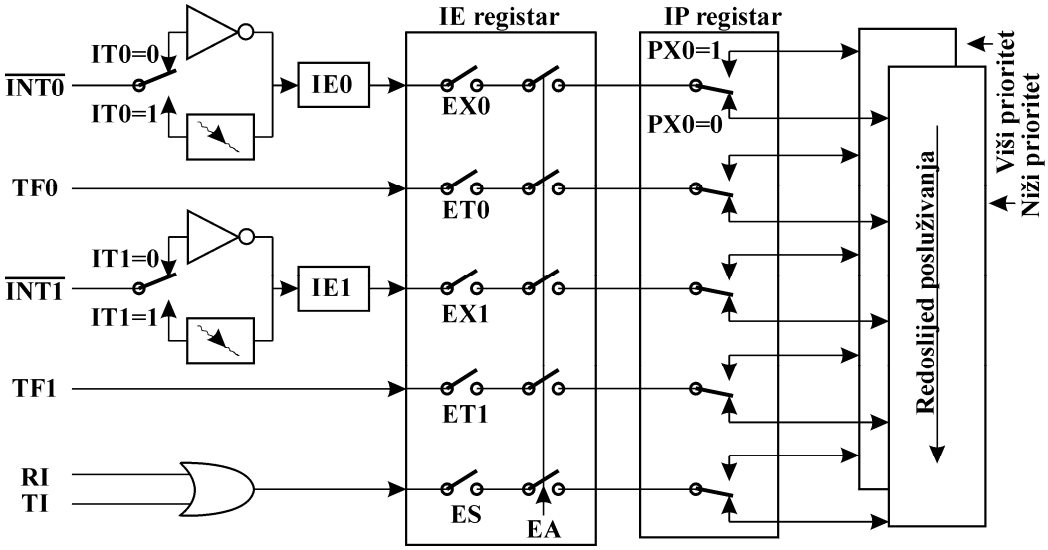
Kao podsjetnik na sklopovlje mikrokontrolera, slike 14.1 i 14.2 prikazuju blokovske sheme brojila T0 i sklopovlja za upravljanje prekidima, zajedno s pripadajućim registrima posebne namjene.



TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TMOD	gate	C/T	M1	M0	gate	C/T	M1	M0
	brojilo T1				brojilo T0			

Slika 14.1 Blokovska shema brojila T0 mikrokontrolera 8051.



IE	EA	X	X	ES	ET1	EX1	ET0	EX0

IP	X	X	X	PS	PT1	PX1	PT0	PX0

Slika 14.2 Blokovska shema sklopovlja za upravljanjem prekidima mikrokontrolera 8051.

15 Završne napomene

Ciklus vježbi i predavanja posvećen mikrokontrolerima sadržava rad s mikrokontrolerom 8051. Rad s ostalim mikrokontrolerima iz familije 8051 praktično se ni po čemu ne razlikuje. Rad s nekim drugim mikrokontrolerom također sadrži sve elemente pokazane na vježbama, pri čemu se od korisnika traži vladanje arhitekturom dotičnog mikrokontrolera.

Ovom prilikom treba naglasiti da izneseno gradivo predstavlja samo osnove. Za rad na stvarnim projektima čitatelju se pored ove literature preporučuje korištenje podataka proizvođača sklopovlja, te podataka proizvođača programskih alata.

Programski paket Keil C51 pored funkcija opisanih na vježbama sadrži i još neke funkcije koje zbog ograničenog vremena nisu obrađivane. Unatoč tome, praktično znanje stečeno na ovom ciklusu laboratorijskih vježbi trebalo bi biti dovoljno za samostalan daljnji rad i učenje kroz projekte.

Literatura

[1] *MCS 51 Microcontroller Family Users Manual*, Intel Corporation, 1994.

[2] *80C51 Family Hardware Description*, Philips Semiconductors, 1996.

[3] *80C51 Family Programmer's Guide and Instruction Set*, Philips Semiconductors, 1997.

[4] *80C51 High Performance ROM 8-bit Microcontroller*, Atmel Corporation, 2002.

[5] *C51 Development Tools, Keil - an ARM Company*, *www.keil.com*.

[6] M. Vučić, *Upotreba mikrokontrolera u ugrađenim računalnim sustavima*, FER, 2007.

[7] M. Vučić, *Ugradbeni računalni sustavi, drugi ciklus - Materijali za predavanja*, 2007, *www.fer.hr/predmet/urs*.

[8] M. Vučić, D. Petrinović, *Projektiranje ugrađenih računalnih sustava - Laboratorijske vježbe*, FER, 2007.

[9] *DS1620 Digital Thermometer and Thermostat, Data Sheets*, Maxim-Dalas Semiconductors, 2003.