

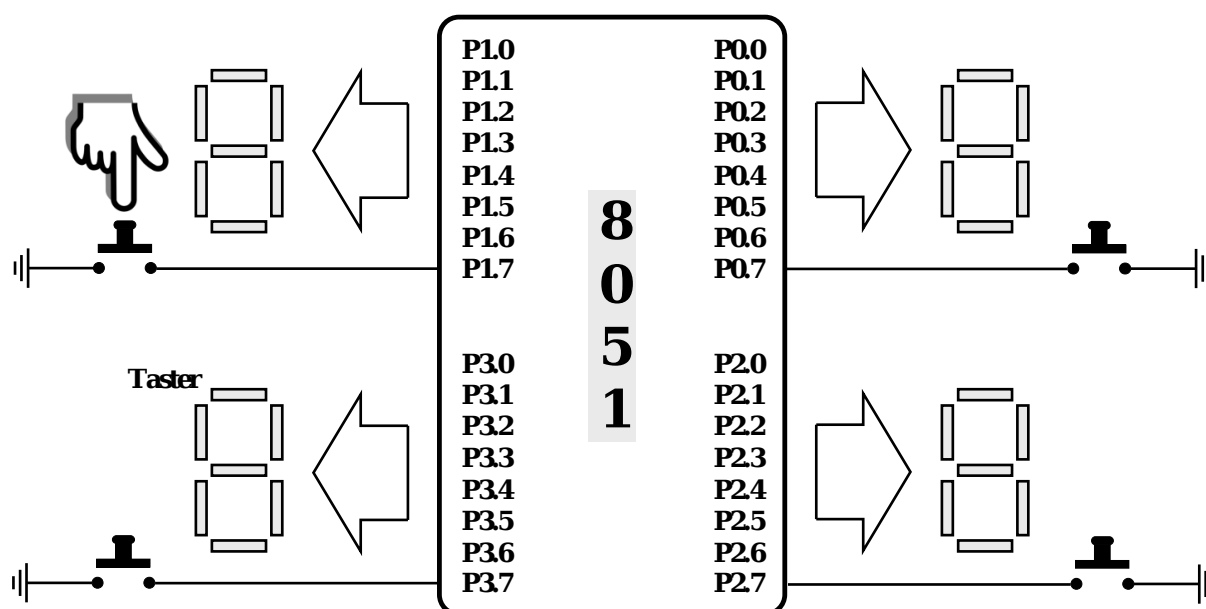
Vežba 7.

Direktno i memorijsko povezivanje periferija sa 8051 mikrokontrolerom

Direktno povezivanje

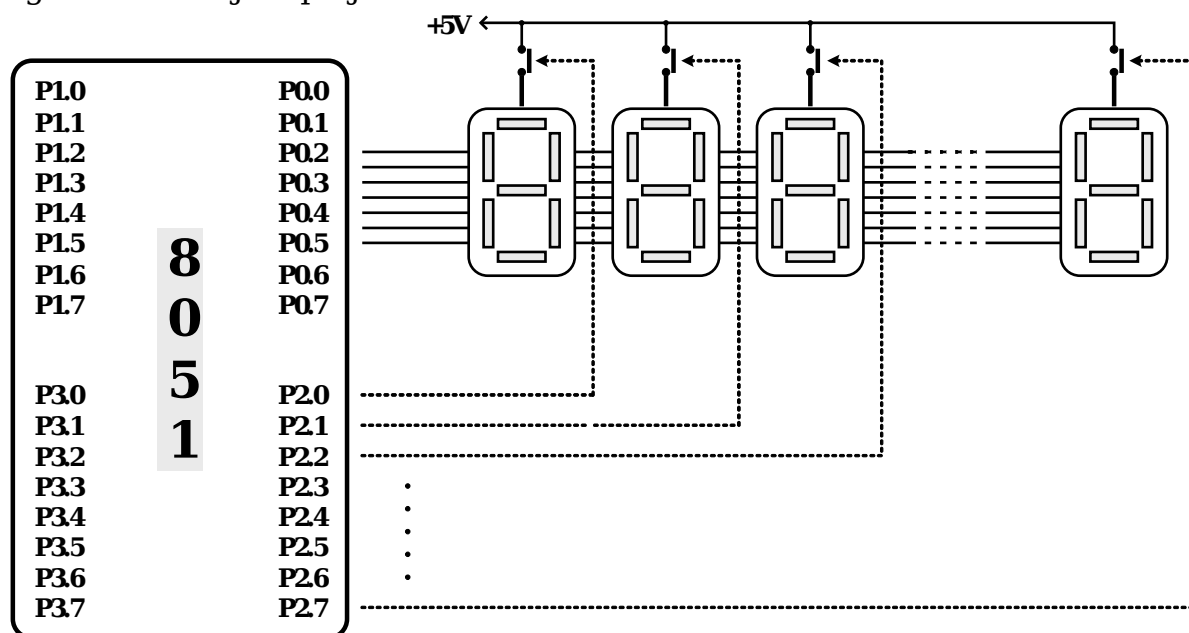
Mikrokontroler 8051 za povezivanje sa periferijama ima dosta mogućnosti, ali je ovo u suštini ograničen sa brojem fizičkih nožica za povezivanje tj. brojem nožica na portovima. Međutim, za kontrolne procese sastavljene od samo nekoliko hardverskih celina kojima treba upravljati mikrokontroler predstavlja pravi izbor. Naravno, podrazumeva se da pomenuti procesi nisu složeni u smislu da ih mikrokontroler ne može obraditi na efikasan način. Periferni uređaji mogu biti npr. svetleće diode (LED), sedmosegmentni LED displeji, prekidači, memorije, drajverska kola za motore, razni digitalni senzori itd. Mi ćemo najčešće raditi sa sledećim periferijama: svetlećim diodama, displejima i prekidačima. Međutim, iako je samo će biti obrađen samo podskup mogućih periferija, nikako se ne gubi na opštosti u radu sa periferijama.

Primeri obrađeni u prethodnoj vežbi (elektronska kockica, rotiranje i blinkanje) predstavljaju direktno povezivanje periferija na mikrokontroler. Međutim, posmatrajmo sledeću hardversku realizaciju (sl. 1).



Slika 1. Potpuna zauzetost svih portova mikrokontrolera kod direktnog povezivanja

Postavlja se pitanje, šta ako želimo da prikažemo i sekunde? Jasno je da je to fizički nemoguće, jer ne postoje fizički nožice portova za proširenje. U ovom slučaju moguće je vezati više displeja multipleksno (sl. 2) uz dodavanje spoljašnjih bafera, ali i u ovom slučaju možemo vezati ograničen broj displeja.



Multipleksno vezivanje se može efikasno koristiti kod displeja, svetlećih dioda ili prekidača, a moguće je čak i displeje i prekidače vezati tako da dele istu magistralu podataka. Međutim, problem je ako treba za mikrokontroler nakačiti puno periferija koje se razlikuju po vezivanju i upravljanju. U ovom slučaju se mora koristiti memorijsko mapiranje.

Ranije je objašnjeno da kućište mikrokontrolera ima 40 nožica i da je proizvođač u cilju obezbeđivanja veće funkcionalnosti čipa implementirao multipleksiranje određenih signala. Ako se koristi mikrokontroler sa internim ROM-om i nema spoljnog ROM-a ili RAM-a, portovi P0 i P2 mogu koristiti sve linije porta kao univerzalni ulazi ili izlazi. Međutim, osim funkcionalnosti poznatije kao univerzalni ulazi i izlazi, ovi portovi mogu biti i drugačije konfigurisani. Tako je port P0 dobio dve funkcije: kao izvor signala adresa A0-A7 i ulaz/izlaz podataka D0-D7. Pre svakog očitavanja programa iz spoljne memorije ili prozivanja RAM-a mikrokontroler na P0 prosleđuje niži bajt adresnog registra i aktivira izlaz ALE. Spoljni registar (najčešće LATCH registar tipa 373 ili 573 iz TTL 74xx familije ili 8282) na visok nivo ALE memoriše stanje P0, a

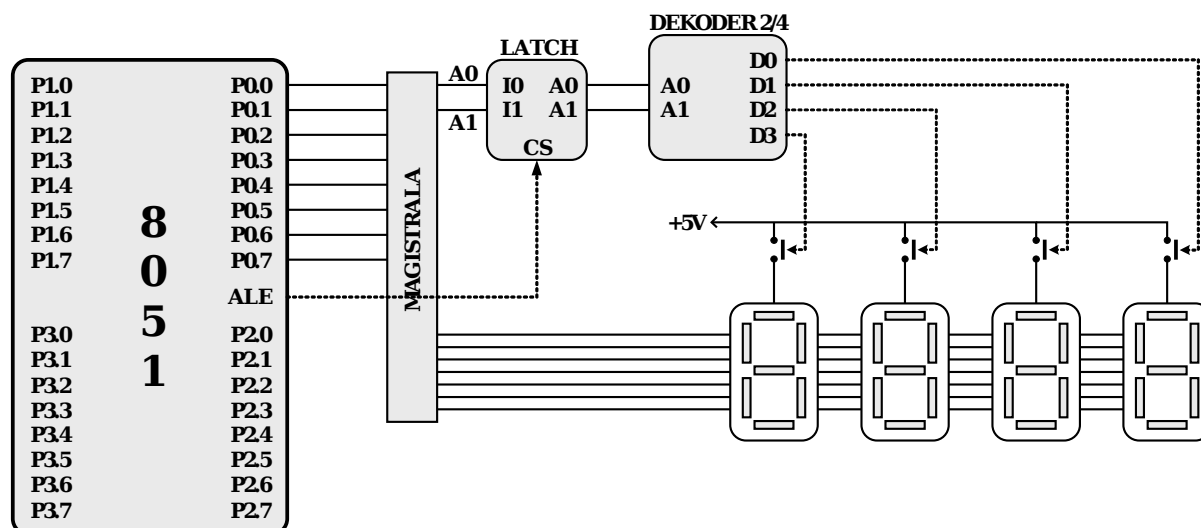
izlazi ovog registra se koriste kao A0-A7. U drugom delu mašinskog ciklusa mikrokontrolera P0 se koristi kao magistrala podataka (Data Bus).

Slično se ponaša i port P2, tj. ako se koristi spoljna memorija, onda su ovo visoki adresni izlazi, od A8 do A15. U tom slučaju, čak i ako se koriste samo neke adrese, preostale nožice ovog porta ne mogu da se koriste kao ulazi ili izlazi.

Dakle, mikrokontroler preko portova P0 i P2 (ukupno 16 adresa) može da adresira 64k memorije. Korišćenjem spoljašnjih komponenti za pamćenje digitalnih vrednosti (latch) iste linije čine i magistralu podataka. Kada su na portovima adrese a kada podaci, govori signal sa specijalne nožice mikrokontrolera označene kao ALE. Iako je memorija praktično periferni uređaj, postavlja se pitanje, sta je sa ostalim periferijama kao što su displeji, prekidači ili svetleće diode?

Kontrolni procesi koji se mogu implementirati 8051 mikrokontrolerom, najčešće ne zahtevaju pun opseg spoljašnje memorije, bilo da je u pitanju RAM, ROM ili kombinacija. To znači da veliki deo memorijskog ostaje neiskorišćen. Ne samo da deo memorijskog prostora ostaje neiskorišćen, već su nožice portova koji nisu važne za formiranje adrese neupotrebljivi. Na primer, ako nam je potrebna eksterna memorija od 1k bajta, za adresiranje je potrebno 10 linija od ukupno 16 sa portova P0 i P2. Ostale nožice, iako ne učestvuju u formiranju adresa, ne mogu biti iskorišćene za vezivanje periferija, jer to ne dozvoljava mikrokontroler.

Ipak, moguće je vezati periferne uređaje ako ih posmatramo kao delove memorijskog adresnog prostora mikrokontrolera. Kombinacijom kola za pamćenje stanja i adresnog dekodera moguće je povezati veliki broj periferija. Maksimalan broj „jednobajtnih“ periferija (npr. displeja) koje je moguće povezati na mikrokontroler je $2^{16}=65536$ uz pretpostavku da imamo dostupan dekodler 16/65536. Način mapiranja se može shvatiti na jednostavnijem primeru za 4 displeja (sl. 3).



Slika 3. Memorijski mapirano povezivanje displeja

Na slici 3 prikazan je jednostavan primer memorijskog mapiranja periferija. Iako se koristi samo port P0 i samo dve fizičke linije za

adresiranje, opisani način adresiranja ne gubi na opštosti u odnosu na slučaj kada se koristi i port P2 tj. sve fizičke linije za adresiranje. U primeru su na mikrokontroler 8051 vezana četiri displeja što znači da je za adresiranje potrebno dva bita, u ovom slučaju to su bitovi sa nožica P0.0 i P0.1. Iste nožice su deo magistrale podataka koju čine sve nožice porta P0. Ranije je pomenuto da su portovi P0 i P2 multipleksirani i da se za razlikovanje adresa od podataka koristi signal sa nožice ALE. Prvo se na magistrali postave adrese i aktivira se signal ALE. ALE signal se vezuje na cs (cs – chip select) ili češće na odgovarajući *enable* ulaz LATCH kola i ako je on visok ulazni signali (I0 i I1) tj. adrese se prosleđuju na izlaz. Na ovaj način se izlazi se pamte sve dok se ne ponovo ne pojavi visok nivo ALE signala.

Kada se ALE signal vrati na nisku vrednost, na magistrali se postavljaju podaci. U međuvremenu dekodirani adresi selektuje memorijsku lokaciju ili registar perifernog uređaja (u ovom slučaju displej) i vrši se upis podataka sa magistrale.

Primeri i zadaci

Elektronska kockica 1. Koristeći mikrokontroler Atmel 89C51 radne frekvencije 12MHz kod koga je na port **P3.0** vezan jedan taster, a jedan 7-segmentni displej je memorijski mapiran na adresi 0x8000. Napisati program koji realizuje elektronsku kockicu (brojanje od jedan do šest). Koristiti timer1, a frekvencija promene brojeva treba da bude 200Hz. Dakle, zadatak je isti kao u prethodnoj vežbi ali se se umesto ledovki koristi displej za prikazivanje rezultata bacanja.

Rešenje:

```
//
// Program koji realizuje el. kockicu
//
#include <reg51.h>
#define FOSC 1000000 // 12MHz/12
#define FR 200
#define DIVIDED -FOSC/FR // ovoliko treba timer da odbroji
#define TMR1_HI ((DIVIDED >> 8) & 0xff)
#define TMR1_LO (DIVIDED & 0xff)
#define SetTimer1() TH1 = TMR1_HI; TL1 = TMR1_LO

typedef unsigned char byte;
// 1, 2, 3, 4, 5, 6
const byte code brojevi[6] = {0x06,0x5B,0x4F,0x66,0x6D,0x7D};

byte xdata displej_at_ 0x8000; //memorijsko mapiranje
sbit taster = P3^0;
bit taster_pritisnut;
byte brojac;

void Inicijalizacija()
{
```

```

        IE = 0x88;           //dozvola prekida tajmera1
        TMOD = 0x10;         //16 bitni tajmerski rezim
        SetTimer1();         //postavka pocetnog stanja tajmera1
        TR1=1;               //start tajmera
        brojac = 0;
        taster_pritisnut = 0;
    }

void Timer1() interrupt 3
{
    TR1 = 0;
    SetTimer1();
    if (++brojac == 6) brojac = 0;
    displej = brojevi[brojac]; // prikazi vrednost na
                               // displeju
    TR1 = 1;
}

void Main()
{
    Inicijalizacija();
    while(1)
    {
        if(taster && !taster_pritisnut)
        {
            taster_pritisnut = 1;
            TR1 = ~TR1;
        }
        if (!taster) taster_pritisnut = 0;
    }
}

```

Program radi isto kao u prethodnoj vežbi. Međutim, umesto ledovki koristi se memorijski mapiran displej na adresi 0x8000. To konkretno znači da prilikom upisa u promenljivu *displej* generisaće se prvo adresa za selektovanje displeja, a zatim će se na magistrali postaviti podaci koji predstavljaju kombinaciju za uključivanje segmenata displeja u skladu sa rezultatom bacanja.

Elektronska kockica 2. Uraditi isti zadatak, ali da pri tome i taster bude memorijski mapiran na adresi 0x8000 (na LSB mestu). Objasniti kako je moguće da i displej i taster budu na istoj adresi?

Brojač nadole 1. Na raspolaganju je mikrokontroler Atmel 89C51 radne frekvencije 12MHz, na koji su povezana 4 sedmosegmentna displeja (memorijski mapirana na adresama 0x8000 do 0x8003) i 4 tastera na portovima **P3.0** do **P3.3**. Na dva displeja se prikazuju minuti, a na preostala dva sekunde. Tasteri služe za resetovanje, početak odbrojavanja, podešavanje početnog stanja minuta (uvećanjem minuta za po jedan) i podešavanje početnog stanja sekundi (uvećanjem sekundi za po jednu).

Napisati program u programskom jeziku C koji omogućava podešavanje početnog stanja vremena, aktivaciju odbrojanja, prikazuje proces odbrojanja na displejima, aktivaciju reseta. Nakon odbrojanog vremena ispisuje na displejima – G O –.

Rešenje:

```
#include <reg51.h>
#define FOSC 1000000          // 12MHz/12
#define TOTAL_CNT 50000      //najveca vrednost koju tajmer treba
                                //da odbroji, a da je fosc celobrojno
                                //deljivo tom vrednoscu
#define TMR0_HI ((-TOTAL_CNT >> 8) & 0xff)
#define TMR0_LO (-TOTAL_CNT & 0xff)
#define SetTimer0() TH0 = TMR0_HI; TL0 = TMR0_LO

#define CRTICA 0x40
#define SLOVO_G 0x7D
#define SLOVO_O 0x3F
#define BROJ_0 0x3F
#define BROJ_PREKIDA 10

typedef unsigned char byte;

const byte code brojevi[10] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x27,0x7F,0x6F};
                                // 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

// memorijski mapirani sedmosegmentni displeji
// na adresama od 0x8000-0x8003
byte xdata displej_minuti1_at_ 0x8003;
byte xdata displej_minuti0_at_ 0x8002;
byte xdata displej_sekunde1_at_ 0x8001;
byte xdata displej_sekunde0_at_ 0x8000;

// dodela imena svakom tasteru
sbit povecaj_minute = P3^0;
sbit povecaj_sekunde = P3^1;
sbit reset = P3^2;
sbit start = P3^3;

bit minuti_taster_pritisnut, sekunde_taster_pritisnut, bio_prekid, provera;
byte sekunde, minuti, brojac_prekida;

void Inicijalizacija()
{
    minuti = sekunde = 0;
    provera = povecaj_minute = povecaj_sekunde = start = reset = 1;
    brojac_prekida = BROJ_PREKIDA;
    IE = 0x82;
    TMOD = 0x01;
    SetTimer0();
    displej_minuti1 = displej_minuti0 = displej_sekunde0 = displej_sekunde1 = BROJ_0;
    TR0=0;
}
```

```

void Timer0() interrupt 1
{
    TR0 = 0;
    SetTimer0();
    bio_prekid = 1;
    TR0 = 1;
}

void Ispis()
{
    displej_sekunde1 = brojevi[sekunde / 10];
    displej_sekunde0 = brojevi[sekunde % 10];
    displej_minuti1 = brojevi[minuti / 10];
    displej_minuti0 = brojevi[minuti % 10];
}

void Resetuj()
{
    TR0 = 0;
    sekunde = minuti = 0;
    provera = 1;
    Ispis();
}

void PovecajMinute()
{
    if(!minuti_taster_pritisnut)
    {
        minuti = (++minuti) % 60;
        minuti_taster_pritisnut = 1;
    }
    provera = 1;
    Ispis();
}

void PovecajSekunde()
{
    if(!sekunde_taster_pritisnut)
    {
        sekunde = (++sekunde) % 60;
        sekunde_taster_pritisnut = 1;
    }
    provera = 1;
    Ispis();
}

void Prekidna_rutina()
{
    if(!(--brojac_prekida))
    {
        if(sekunde) // sekunde > 0?
            sekunde--;
        else
        {
            minuti--;
            sekunde = 59;
        }
    }
}

```

```

        }
        brojac_prekida = BROJ_PREKIDA;
        Ispis();
    }
    bio_prekid = 0;
}

void Go()
{
    TR0 = 0;
    displej_minuti1 = CRTICA;
    displej_minuti0 = SLOVO_G;
    displej_sekunde1 = SLOVO_O;
    displej_sekunde0 = CRTICA;
}

void Main()
{
    Inicijalizacija();
    while(1)
    {
        if(bio_prekid) Prekidna_rutina();
        if(start && provera)
        {
            SetTimer0(); TR0 = 1; provera = 0;
        }
        if(reset) Resetuj();
        if(povecaj_minute) PovecajMinute();
        else
        {
            minuti_taster_pritisnut = 0;
            if(povecaj_sekunde) PovecajSekunde();
            else sekunde_taster_pritisnut = 0;
        }
        if(!(sekunde||minuti||provera))
            Go();
    }
}

```

U ovom primeru je ispis na sedmosegmentne displeje urađen korišćenjem tzv. *memorijskog mapiranja*. Ova metoda omogućava korišćenje perifernih jedinica koje nisu povezane direktno na portove. Suština je u tome da sa stanovišta mikrokontrolera, displeji predstavljaju „memoriju“ u koju mikrokontroler upisuje. Pri radu sa memorijom mikrokontroler koristi portove P0 i P2 za postavljanje 16 bitne adrese (u ovom primeru su to adrese od 0x8000 do 0x8003), a potom se na portu P0 pojavljuje i podatak koji se prikazuje na odgovarajućem displeju na osnovu dekodovanja adrese. Prekidna rutina služi da odredi vreme trajanja jedne sekunde. Procedure PovećajMinute i PovećajSekunde omogućavaju podešavanje početnog vremena za odbrojanje. Procedura Go poziva se kada je odbrojano početno vreme.

Brojač nadole 2. Uraditi prethodni zadatak tako da i tasteri budu memorijski mapirani na adresi 0x8000.

Koristeći 4 sedmosegmentna displeja koji su memorijski mapirani na adresama 0x8000, 0x8001, 0x8002 i 0x8003 i 4 prekidača koji su mapirani na adresi 0x8000 na nižih 4 bita, napisati program u programskom jeziku C koji realizuje:

Blink reči. Pomoću prva 2 prekidača se bira jedna od 4 reči, a pomoću druga dva prekidača se bira frekvencija treptanja te reči. Frekvencije su 5, 10, 20, 40 Hz.

Trčeca reklama. Treba realizovati trčecu reklamu gde je tekst dat kao niz od 16 karaktera. Pomoću dva prekidača se bira brzina kretanja slova (5, 10, 20, 40 Hz), pomoću 3. se bira smer kretanja slova, a 4. prekidač služi za test svih segmenata (ako je on aktivan svi segmenti svetle).