S.T.Š. „Nikola Tesla“
Sremska Mitrovica

Maturski rad

iz predmeta:

Racunari

Tema:

Eksperimentalno kolo na USB portu

Profesori:

dipl. ing. Djoko Krsmanović
dipl. ing. Zeljko Cipar
dipl. ing. Vlada Birovljev

Učenik:

Petrović Strahinja 4e2
Elektrotehnicar racunara

Sadrzaj:

1.UOPSTENO O UPRAVLJANJU PROCESIMA POMOĆU RAČUNARA

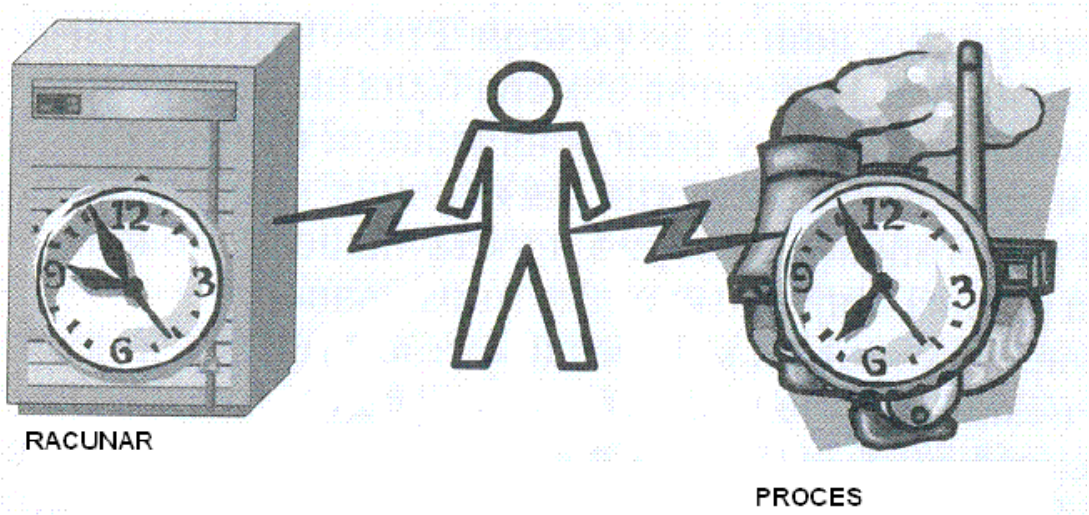
Gledano potpuno uopšteno u svakom procesu postoji neki ulaz(informacija, materijal, signal...) koji se menja unutar njega, i napušta ga u izmenjenoj formu(izlaz procesa). Pošto nijedan proces nije savršen uvek postoji neka korekcija ili izmena u cilju usavršavanja tog procesa. To se moze postići smanjenjem vremena, energije i sl... da bi se ostvario željeni rezultat projektuje se upravljački sistem koji za cilj ima da menja procesne promenljive u cilju poboljšavanja performansi procesa. Tako formiran sistem se zove sistem automatskog upravljanja(SAU).

Zbog potreba upravljanja takvim sistemom javlja se ideja da se za upravljanje koristi racunar. Upravljanje sistemom obuhvata merenje, određivanje upravljanja i izvršavanja komandi.

Postoje dva nacina vezivanja procesa i računara.

Prvi je direktno gde je računar direktno vezan za proces tj računar upravlja sistemom u realnom vremenu. U tom slucaju računar mora da prihvata merene veličine sa procesa, koje se prethodno moraju pretvoriti u digitalni signal, koji je prihvatljiv računaru. Zatim računar mora da obradi merene velicine i na osnovu njih da da izlaz, koji se iz digitalnog oblika mora pretvoriti u pogodan oblik za delovanje na process.

Druga mogućnost korišćenja računara u upravljanju procesima je da se računar koristi samo za izračunavanje. A rezultati merenja dobijenih na procesu se ručno unose u računar, zatim ih računar obradjuje na matematičkom modelu, pa se prema dobijenim rezultatima ručno koriguje proces.



Real time upravljanje u odnosu na offline je mnogo brže, ali ono zahteva da računar stalno bude vezan na proces, i da je u realnom vremenu koriguje svaku nepravilnost u procesu.

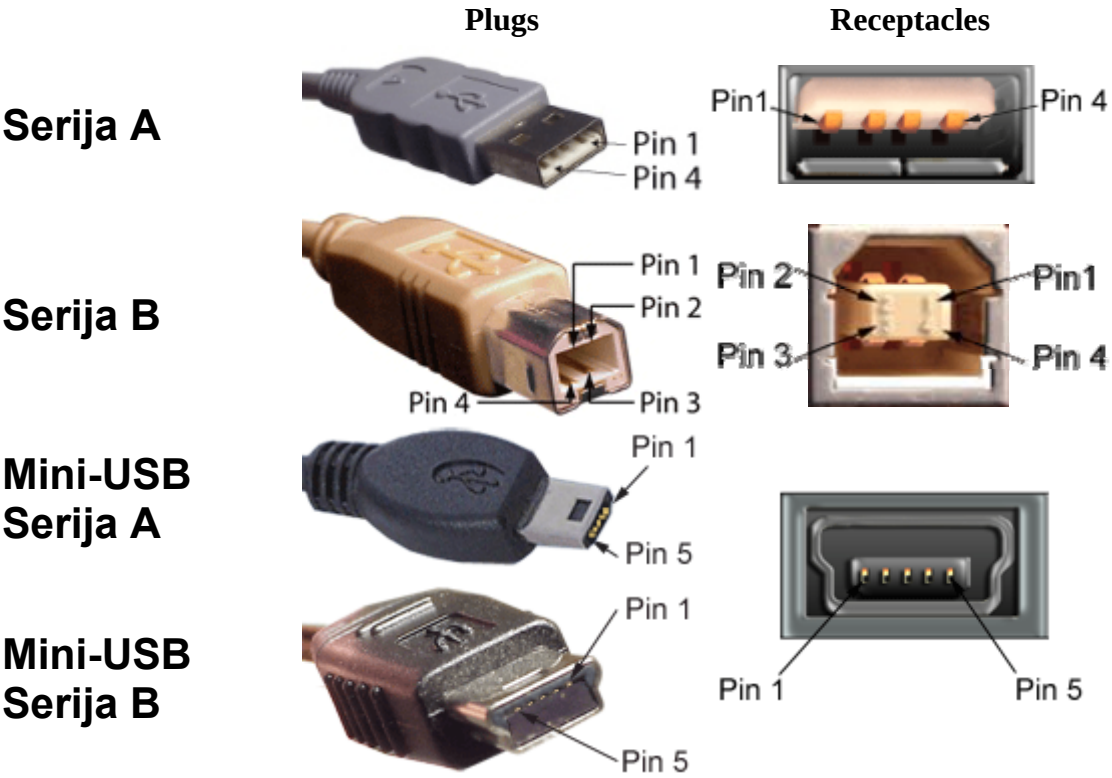
Ovakav sistem moze da reaguje u odredjenim vremenskim intervalima. Tj računar posle određenog vremenskog intervala proverava stanje na procesu, i koriguje ga ako ima potrebe. Drugi način je da računar reaguje na dogadjaj iz procesa. Npr ako je postignuta odgovarajuća temperatura da se isključi grejač, ili ako je postignut odgovarajući nivo da se zatvori ventil i sl.

2. UOPŠTENO O USB PORTU

Universal Serial Bus (USB) je spoljni konektor (priključak) za razne uređaje kao što su štampač, miš, tastatura, digitalna kamera, modem. Pojavio se 1996. godine, a zaživeo tek 1998. Karakteriše ga visoka brzina i jednostavnost instaliranja uređaja, pogotovu sa novijim verzijama operativnih sistema (Plug & Play princip).

USB 1.1 standard je stari standard (iako se još koristi) koji ima brzinu od 12Mbps (megabit po sekundi, 1 Bajt = 8 Bita), a USB 2.0 standard, koji se pojavio 2000. godine, podnosi brzine do 480Mbps koji je već u dobroj meri zamenio USB 1.1 standard, ali i serijske i paralelne priključke.

Raspored pinova na priključcima I konektorima



Standardni USB raspored pinova I boja žica

Pin	Boja žice	Funkcija
1	Crvena	V BUS (+5V)
2	Bela	D-
3	Zelena	D+
4	Crna	Masa

Mini USB tipa A raspored pinova I boja žica

Pin	Boja žice	Funkcija
1	Crvena	V BUS (+5V)
2	Bela	D-
3	Zelena	D+
4	Povezan sa pinom 5	ID
5	Crna	Masa

Mini USB tipa B raspored pinova I boja žica

Pin	Boja žice	Funkcija
1	Crvena	V BUS (+5V)
2	Bela	D-
3	Zelena	D+
4	Nije povezan (*)	ID
5	Crna	Masa

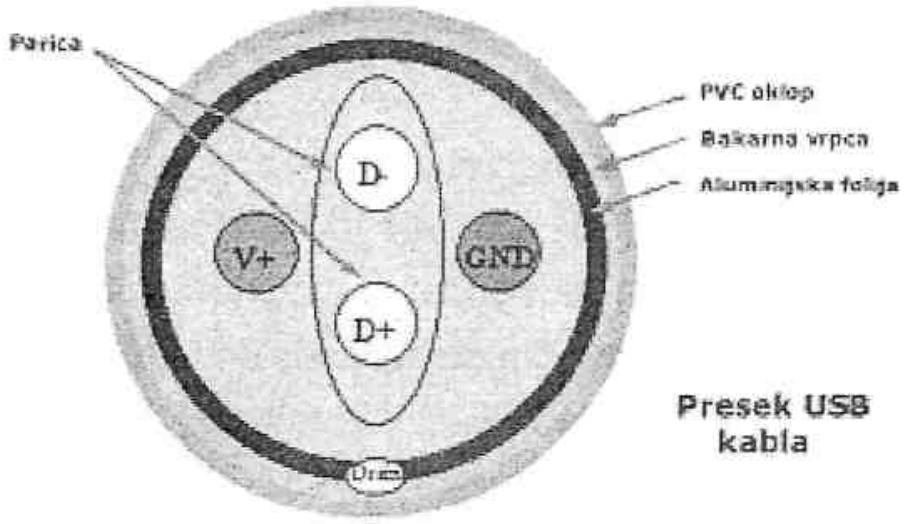
(*) Ponekad vezan za peti pin preko otpornika

3. USB KOMUNIKACIJA

Prvi PC racunari su koristili tri porta, i to: jedan za stampac (LPT) i dva serijska (COMl I COM2), pri cemu je jedan od njih vecinom bio zauzet od strane serijskog misa. Medjutim, njihova sve veca primena je rezultovala pojavom razlicitih uredjaja koji se na njih ukljucuju, kao sto su skeneri, digitalne fotoaparati i slicno. To je uzrokovalo uvodjenje USB porta (Universal Serial Bus), Njegovim uvodjenjem se nastojalo se resavanje sledecih problema:

- Resenje u vezi ogranicenja broja slotova na osnovnoj ploci, kao i broja portova PC racunara.
- Jednostavno prosirenja PC racunara upotrebom softverskih drajvera.
- Mogucnost napajanja eksternih uredjaja koji malo trose od strane racunara.
- Omoguceno je prikljucenje do 127 eksternih uredjaja na glavni USB port, pa se
- time resava ogranicenje koje je pre postojalo: jedan uredjaj - jedan slot.
- Omogucene su velike brzine prenosa do 478 Mb/s.
- Pojednostavljuju se kablovi za prikljucenje uredjaja a njihova duzina se
- povecava.
- Omogucena je kontrola potrosnje eksternih uredjaja.
- Podrzano je autokonfigurisanje ovih uredjaja po principu PnP detekcije novih uredjaja u racunaru.

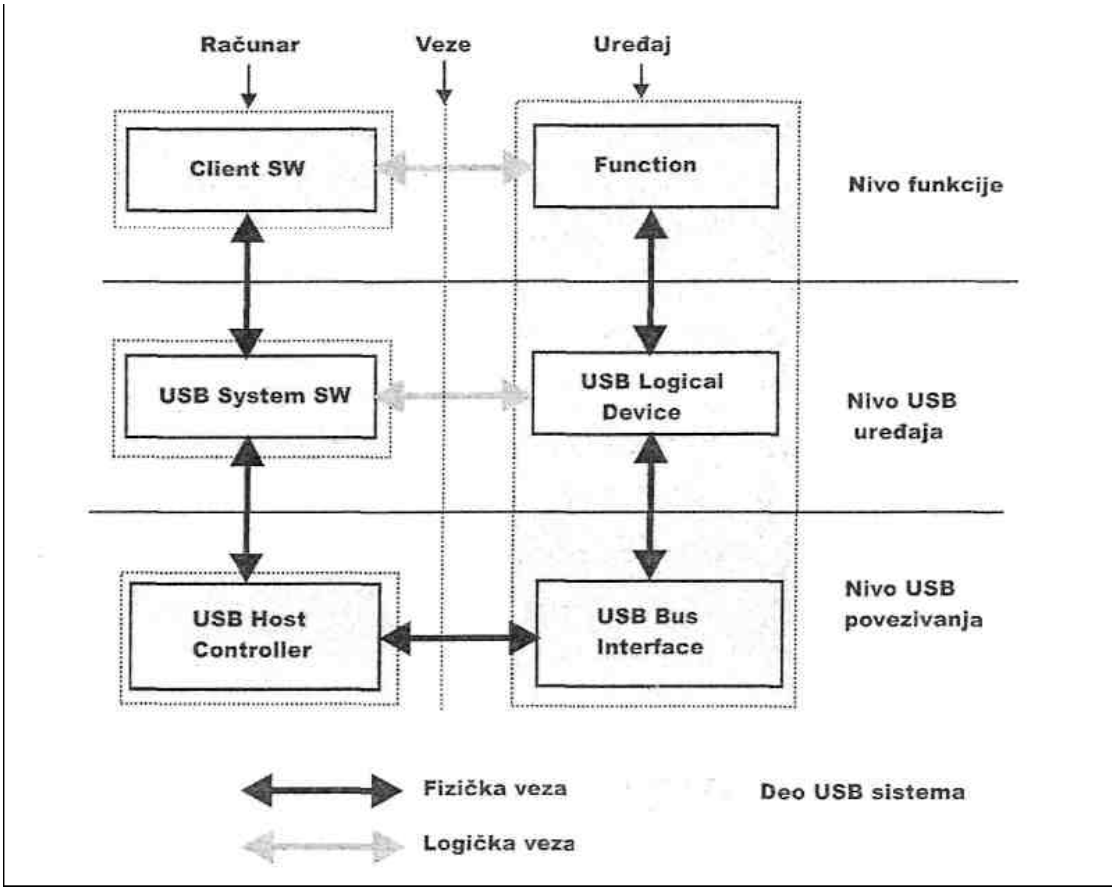
USB standard defmisan je 1996 godine kao verzija 1.0, a koja je dopunjena 1998 godine kada je nastala verzija 1.1. Standard 1.1 podrzava dve brzine prenosa, i to: punu brzinu kada se prenos krece do 12 Mb u sekundi, i za sporije uredjaje nizu brzinu prenosa do 1.5 Mb u sekundi i omogucuje napajanje uredjaja koji ne trose vise od 500rnA. U aprilu 2000. godine dat je novi standard USB verzije 2.0 koje podrzava velike brzine prenosa do 480 Mb u sekundi.
Za prikljucak na USB port koristi se cetvero zicni kabl preko koga se prenose podaci I napajanje sa impedansom 90 Ω, sto je prikazano na slici 1.



Sl. 1 Presek USB kabla

Za prenos podataka se koriste dve linije D+ i D-. Kada se prenosi logicka nula linija D- je na vecem potencijalu od linije D+, dok je u slucaju prenosa logicke jedinice obrnuto. Predajnici moraju da daju napon veci od 2.8 V sa opterecenjem od 15 kΩ. Prijemnici moraju da imaju simetrican ulaz pri cemu do promene stanja dolazi ako se na ulazu prijemnika naponi razlikuju za vise od 200 mV. Svaka linija za prenos podataka ima i nesimetrican prijemnik za detekciju greske koja se pojavi ako su obe linije podataka nadju na istom naponu. Kada se radi o brzom USB prenosu tada se na liniju D+ prikljucuje opteretni otpornik (pull-up) reda 1.5 kΩ, dok se kod sporog prenosa on ukljucuje na liniju D-. Na izlazima drajvera linija D+ i D-prikljucuju se opteretni otpornici reda 15 kΩ. U slucaju kada USB eksterna jedinica nije pod naponom i kada je drajverski izlaz porta u stanju visoke impedanse preko ovih otpornika se odredjuje brzina prenosa. Preko njih se takodje odredjuje da li je eksterni uredjaj povezan na USB cvor. Kada uredjaj nije povezan na USB cvor njegovi izlazni drajveri ce biti u stanju visoke impedanse i obe linije bice na potencijalu mase, koje se naziva nesimetricna nula SEO (*Single Ended 0*). Povezivanjem uredjaja na cvor on ce dobiti napajanje, ali njegovi izlazi i dalje ce biti u stanju visoke impedanse, dok ce napon na liniji podataka porta koji je povezan na opteretni otpor postati visok, sto moze da detektuje cvor. U slucaju da se ne salju USB paketi, linije podataka se nalaze u stanju visoke impedanse.

Organizacija USB magistrale prikazana je na slici 11.



Sl. 11 Organizacija USB magistrale

Iz prikazane slike se vidi da se USB magistrala moze predstaviti sa tri nivoa, i to:

- Nivo USB uredjaja (*USB Interface Layer*) koji obezbedjuje fizicku vezu za prenos signala i paketa izmedju racunara i USB uredjaja. Ovaj nivo omogucuje sistemskim programima koji kontrolisu USB magistralu, rad sa opstim funkcija koje koristi USB uredjaj
- Nivo funkcije se ostvaruje upotrebom klijentskog softvera (*Client SW*)
- Nivo interfejsa obezbedjuje da se fizicki obavlja komunikacija.

Sistem USB povezivanja se deli na cetiri funkcionalno zaokruzene celine:

- USB uredjaj (*USB PhysicalDevice*).
- Klijentski softver (*Client Sofhvare*)
- USB sistemski softver (*USB System Software*)
- Glavni USB kontroler (*USB Host Controller*).

USB predstavlja neki eksterni uredjaj koji se prikljucuje na USB i koji izvrsava zahtevane funkcije. Klijentski softver omogucava da se izvrsi prenos podataka izmedju eksternog USB uredjaja i racunara, a najcesce ga isporucuje proizvodjac USB uredjaja. USB sistemski softver je deo operativnog sistema za podrsku USB uredjaja i isporucuje se uz operativni sistem. Glavni USB kontroler zaokruzuje u jednu celinu hardver i softver koji omogucuje rad USB uredjaja. USB uredjaji sa racunarom komuniciraju slanjem paketa. Na pocetku slanja paketa linija za prenos podataka se postavlja u suprotno stanje od onog u kome se nalazi u mirnom stanju, dok se na kraju paketa podataka, ova linija postavlja u SEO stanje u duzini trajanja od dva bita.
USB uredjaji se mogu resetovati na nekoliko nacina od koji se cesto koristi pristup kada se linija podataka postavi u SEO stanje u trajanju od l0 ms. U slucaju da se USB magistrala nalazi u mirnom stanju vecem od 3 ms tada eksterni USB uredjaji mogu preci u stanje male potrosnje, ako to podrzavaju. Vracanje u radno stanje treba da traje najduze 20 ms. Prenos podatka se vrsi upotrebom NRZI metoda kodiranja (*Non Return Zero Invert*), sto znaci da ako se pojavi logicka jedinica ona ce trajati celom svojom duzinom, odnosno nece doci do promene naponskog nivoa. U slucaju pojave povorki nula napon sa linije se menja za svaki bit, sto se koristi za uspostavljanje signala takta na prijemu. Kada se u povorci podataka pojavi sest uzastopnih jedinica radi sigurnosti na prijemu, vrsi se umetanje bita, tj. ubaci se jedna nula koja se na prijemu izbacuje. Preko USB magistrale se prenosi i napon napajanja V+ koji iznosi + 5 V, uz maksimalno opterecenje do 5 A, pri cemu potrosnja pojedinacne eksterne jedinice ne sme preci 400 mA kada je u radnom stanju, a u stanju mirovanja 500 μA. Ovaj uslov ne mogu ispuniti svi USB uredjaji tako da u tom slucaju moraju koristiti sopstveno napajanje. Prilikom inicijalizacije USB sistema po ukljucenju racunara on pribavlja podake o svim uredjajima koji povezani na
USB magistralu radi numerisanja magistrale (*bus enumeration*). Za ispravan rad USB uredjaja koji su povezani na racunare svaki od njih mora da ima adresu koje se krecu u opsegu od 0 do 128 i dodeljuje ih racunar prilikom konfigurisanja magistrala. Adresa 0 je adresa koju koristi

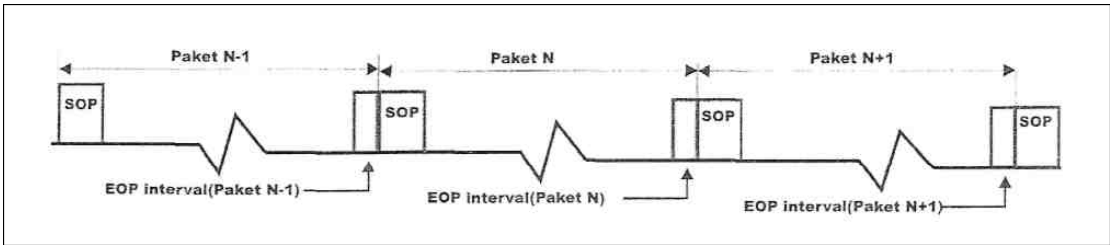
racunar za postavljanje uredjaja povezanih na USB magistralu i ne mogu je koristiti eksterni uredjaji.

3.1. USB protokoli

Svaka USB transakcija je sadrzana od:

- Token Paketa (njegovo zaglavlje definise sta ce dalje slediti)
- Opcionalnog Data Paketa (sadrzi podatke)
- Status Paketa (koristi se u kontroli transakcija i omogucuje korekciju greski)

Host inicira sve transakcije. Prvi paket je token i omogucuje hostu da opise sta ce dalje slediti i koje ce biti vrste transakcija podataka tj. citanja ili upisivanja kao i koja je adresa uredjaja i odgovarajuci endpoint. Sledeci paket je generalno paket podataka iza koga sledi handshaking paket ako je Data paket ispravno primljen, odnosno Stall paket ako to nije slucaj. Obicno se paketi sastoje od sledecih polja: Pocetak svakog paketa pocinje sinhronizacionim bajtom SOP(SYNC), dok se sledeci paket razdvaja od prethodnog sa EOP sto je prikazano na slici 12.



Sl.12 Prikaz slanja podatka na USB magistrali

Na pocetku svakog paketa salje se sinhronizacioni bajt (SYNC) koji se sastoji od sedam nula i jedne jedinice (80H) za spore USB jedinice i 32 bita za USB jedinice sa punom i velikom brzinom prenosa. Na osnovu ovog bajta sinhrono kolo na strani prijemnika generise taktni signal.
Nakon sinhronizacionog bajta sledi polje za identifikaciju paketa PID (*Packet Identifier*), kod koga se prva 4 bita koriste za identifikaciju vrste paketa, dok su sledeca cetiri bita (vece tezine) invertovani bitovi PID**-a,** na osnovu cega se proverava tacnost primljenog PID-a. Polje PID definise vrstu paketa i

njegov format, kao i tip detekcije greske. U tabeli 1 prikazane su PID vrednosti za pojedine grupe paketa.

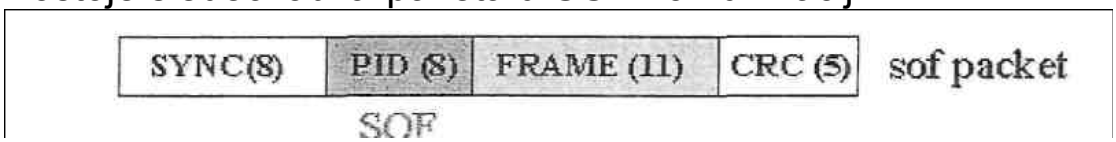
Grupa paketa	PID Vrednost	Vrsta paketa
Token	0001	OUT *Token*
	1001	IN *Token*
	0101	SOF *Token*
	1101	SETUP *Token*
Data	0011	DATA0
	1011	DATA1
	0111	DATA2
	1111	MDATA
Handshake	0010	ACK *Handshake*
	1010	NAK *Handshake*
	1110	STALL Handsake
	0110	NYET(Ne odgovara Yet)
Special	1100	PRE*amble*
	1100	ERR
	1000	Split
	0100	Ping

Tabela 1 PID vrednosti pojedinacnih vrsta paketa

Adresno polje (ADDR) je duzine 7 bita i odredjuje jedinicu kojoj se paket salje, pa je moguce ukupno adresirati 127 jedinica, s tim sto adresa 0 nije dozvoljena, dok jedinice koje nisu adresirane moraju slati paketa sa adresom 0.
Endpoint polje (ENDP) sadrzi 4 bita, omogucujuci 16 endpointa. Spore USB jedinice koriste samo 2 dodatna endpointa na vrhu podrazumevanog kanala (4 endpointa su maksimalna). *Cyclic Redundancy Checccs*(CRC) koristi se za kontrolu ispravnosti prijema paketa. Svi token paketi imaju 5-bitni CRC dok Data paket ima 16-bitni CRC. Kraj paketa (ENDP), signalizira se pomocu SEO stanja.

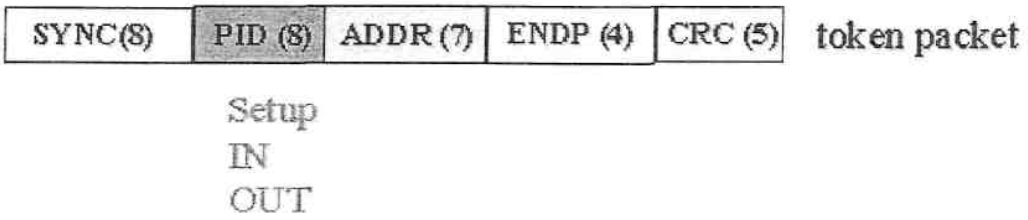
3.2. USB tipovi paketa

Postoje sledeci oblici paketa u USB komunikaciji:



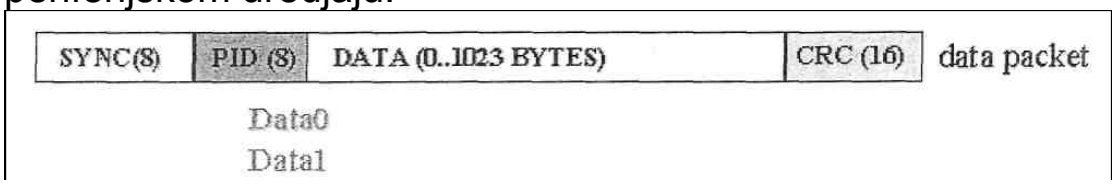
Sl. 13 Format SOF paketa

SOF (Start of Frame Packets) paket ciji je PID=0101 salje 11 - bitni podatak o broju okvira *(*frame*),* kao i 5 bitova CRC detekcije. Ovaj paket salje host svake milisekunde ±500 ns na USB jedinicama sa punom brzinom ili svakih 125μs±0.0625μS na brzim USB jedinicama.



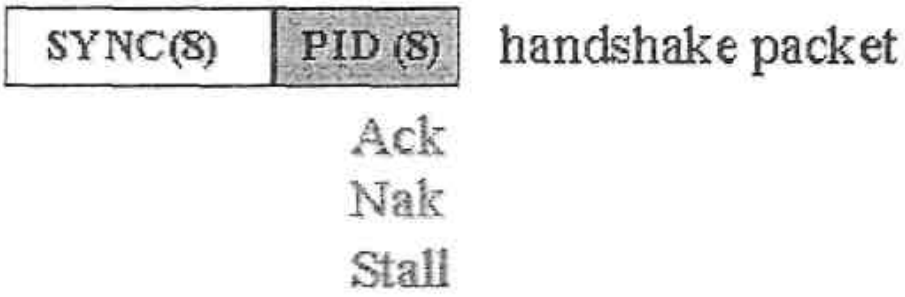
Sl.14. Format Setup, IN, Out paketa

Setup, IN i OUT paketi imaju oblik kao na slici 14. Setup paket ciji je PID=1101 vrsi setovanje funkcije od strane racunara i sadrzi tekucu i krajnju (Endpoint) adresu. IN paket ciji je PID=1001 je prvi paket od eksternog uredjaja prema racunaru. OUT paket ciji je PID=0001 je prvi paket koji racunar salje periferijskom uredjaju.



Sl.15. Format Data paketa

Data0 paket ciji je PID=0011 je paran paket podataka koji sadrzi do 1023 bajta podataka. Datal paket ciji je PID=1011 je neparan paket podataka. Ovi paketi za CRC kontrolu koriste 16 bita.



Sl.16. Format Ack, Nak, Stall paketa

Ack paket koji ima PID=0010 je potvrda prijemnika da je paket primljen bez greske. Nak paket koji ima PID=1010 je odgovor predajnika o neispravno primljenom paketu. Stall paket koji ima PID=1110 pokazuje da je neko odrediste zaguseno, tj. ne moze da primi sve do tada poslate pakete. Prenos podataka pocinje tako sto racunar posalje paket u kome je definisan tip i smer
prenosa, adresa USB uredjaja i adresa krajnjeg odredista. Ovaj paket se naziva *token.* Adresirani uredjaj detektuje svoju adresu iz adresnog polja i time bude selektovan i postaje spreman za prijem ili predaju podataka. Primalac odgovara slanjem odzivnog paketa (handshake packet) izvestavajuci o uspesnosti prenosa. Adresiranje uredjaja se vrsi preko adresnog polja paketa od 7 bita. Adresa se koristi kod IN, OUT ili Setup paketa. Paket SOF sadrzi broj frejmova od 11 bita, pri cemu se njegov sadrzaj inkrementira za svaki novi okvir sve do vrednosti 7FFH. Paketi podatataka Data0 i Data1 mogu da sadrze do 1023 bajta i to
koriste USB brze jedinice, dok kod sporih jedinica ta je duzina 8 bajta. Ova dva paketa omogucavaju jednostavnu sinhronizaciju predajnika iprijemnika u slucaju zahteva predajnika za ponovnim slanjem radi greske u prenosu. Predajnik salje novi paket tek kada od prijemnika dobije porvrdu o uspesnosti, tj. paket Ack. Prvo se salje paket Data0, pa paket Datal i tako naizmenicno. Svi paketi imaju na kraju bitove za redundantu proveru greske (CRC) da li je paket ispravno primljen ili nije. Pri prenosu podataka uvek se obavlja komunikacija u oba smera i uvek je inicijalizirana od strane racunara. Smer prenosa definise racunar slanjem IN ili OUT paketa. U IN paketu racunar zahteva od eksternog USB

uredjaja da mu posalje podatke. Nakon ovog paketa eksterni uredjaj ili racunar salje Data paket, pri cemu se na kraju svakog primljenog paketa strana koja je vrsila prijem salje potvrdu uspesnosti slanjem Ack, Nak ili Stall paketa.

3.3. USB funkcije

Najcesce se misli na USB jedinicu, ili USB periferiju ali osim njih postoji i USB jedinica za slanje, koju koristi host ili periferija, odosno USB Hub ili Host kontroler, ili USB periferijska jedinica. Zbog toga su uradjene odgovarajuce USB funkcije koje omogucuju USB jedinicama sposobnost i funkcionalnost kao sto su stampaci, skeneri, modemi i druge periferijski uredjaji.
Najveci broj USB funkcija podrzava USB protokole niskog nivoa sve do transakcionog sloja. Razlog pokrivanja vecine USB funkcija je zato sto ce kontroleri prijavljivati greske kao u slucaju PID encoding greske. Vecina funkcija ima seriju bafera obicno duzine 8 bajta. Svaki bafer ima odgovarajuci endpoint - EPO IN, EPO OUT itd. Pretpostavimo da host posalje zahtev za Device deskriptorima. Funkcija hardvera procitati ce Setup paket, odredjujuci iz adresnog polja da li je paket za njega, i ako je tako on ce iskopirati podatke iz paketa u odredjeni endpoint bafer odredjen vrednoscu polja Setup token-a. Tada ce slediti slanje Handsake paketa od primaoca i generisanje internog interapta unutra mikrokontrolera za odgovarajuci endpoint nakon prijema paketa. Softver odmah obradjuje interapt i cita sadrzaj endpoint bafera i izdvaja device deskriptore.

3.4. Endpoints

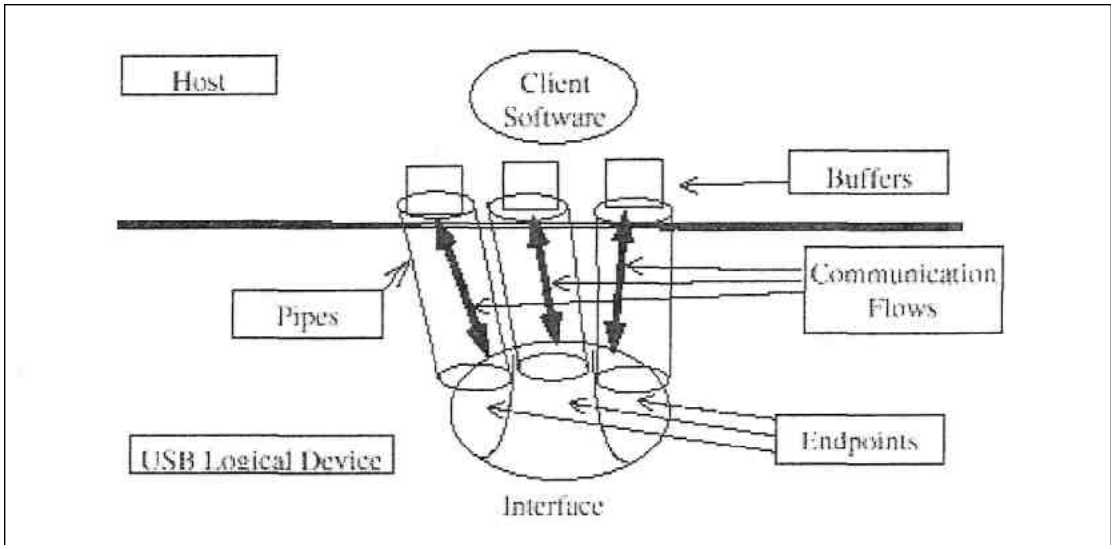
Endpoints se moze opisati kao izvor podataka. Kako je bas centar hosta, to se endpoint pojavljuje kao kraj komunikacionog kanala u USB funkciji. Softverski sloj, odnosno drajver jedinice moze slati pakete jedinici npr. EPI. Podaci koji izlaze iz hosta, kao kraj imace EP OUT bafer. *Firmware* u USB jedinici moze tada procitati podatke. Ako se radi o vracanju podataka, funkcija ne moze jednostavno upisati podatke na bas jer je on kontrolisan od strane hosta.

Zato se podaci upisuju u EPI IN bafer koji u njemu ostaju sve dotle dok host ne posalje zahtev za IN paketom od endpointa od koga su zahtevani. Zbog toga se endpoint moze smatrati interfejsom izmedju funkcija hardvera i firmware koji obradjuje funkcije jedinice. Sve jedinice moraju podrzavati nulti endpoint. To je endpoint koji primaju sve kontrole jedinice i statusne zahteve u toku enumeriacije sve dotle dok jedinica ne bude operativna na bas-u.

3.5. Pipes

U isto vreme jedinice salju i primaju podatke iz razlicitih *endpoint*-a, a pomocu *Client* softvera kroz razlicite *pipes*. *Pipe* je logicka konekcija izmedju *hosta*-a i *endpoint*-a. Pipes predstavlja skup parametara koji odredjuju sirinu njegovog opsega, koji je tip transfera (*Control, Bulk, Iso* ili *Interrupt),* smera kretanja podataka kao i maksimalnu velicinu paketa/bafera. Podrazumevani pipe je bi-direkacionalni, uradjen tako da omogucuje kako nulti endpoint ulaz tako i nulti endpoint izlaz sa Control transfer tipom. Pipe-ovi se dijele na pipe-ove za:

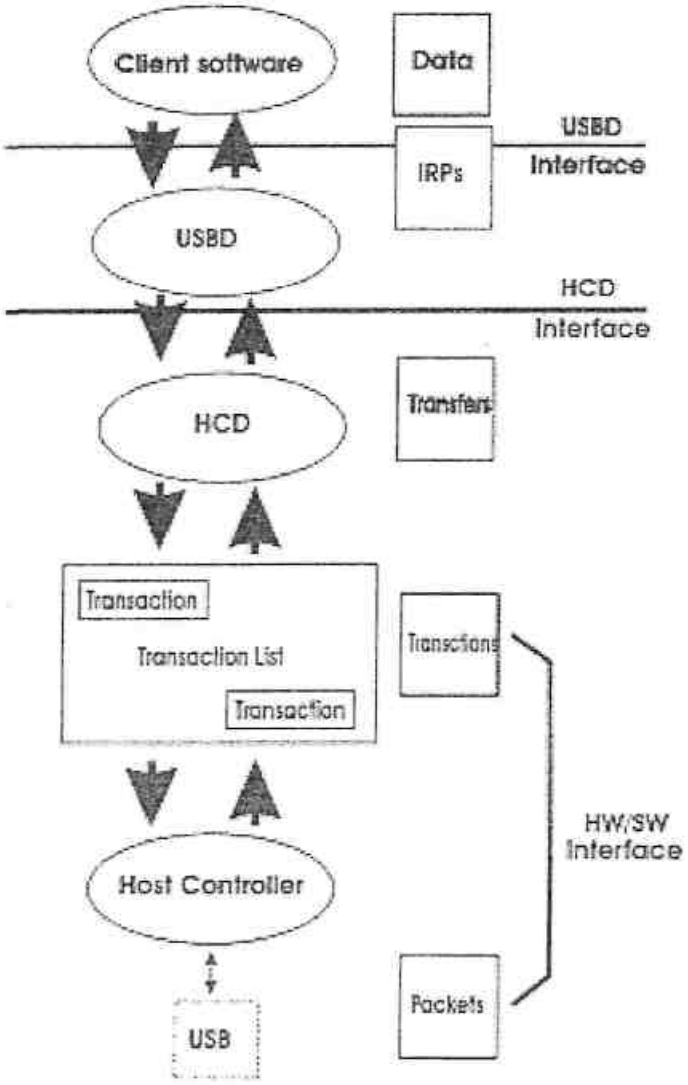
- tokove (*stream*) - za *interrupt*, *bulk* i *isochronus* prenos
- poruke (*message*) - za kontrolni (*control*) prenos (za *default pipe*)



Sl. 17 Tokovi komunikacije u USB sistemu

3.6. HID protokoli

Univerzalni serijski bas (USB) povezuje se preko habova i funkcija sa PC-jem. Habovi obezbedjuju tacke povezivanja, u isto vreme sa funkcijama za pristup PC-u. HID *Interface* je funkcija koja pripada tzv. *Human Interface Device* (HID) i podklasa je od USB komunikacione arhitekture.
USB arhitektura sastoji se od 4 sloja i to: clinet sofhvare, USB driver (USBD), *Host Conrollller driver (HCD) i Host Controllera(HC)* sto je prikazano na slici 18.

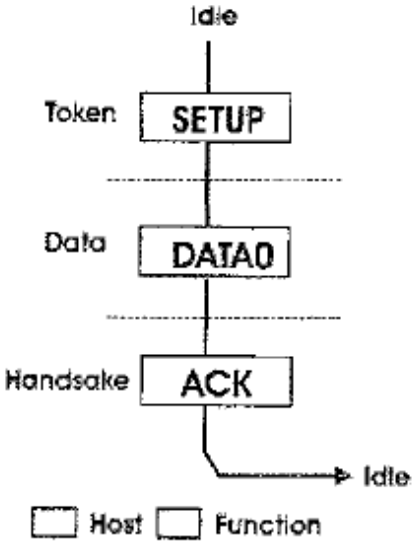


Sl. 18 USB konverzija informacija od Client Software ka Bus-u

Client software upotrebljava/generise specificne funkcije za podatke u/iz a funkcija endpoint preko poziva i povratnog poziva zahteva IRP (*Input*/*Output* Request Packets) sa USB interfejsom. USBD konvertuje podatke u clientTRPs u/iz jedinici kao krajnjoj tacki preko poziva/povratnog poziva sa odgovarajucim HCD. HCD konvertuje IRP u/iz transakcije I organizuje ih za manipulaciju pomocu Host Controller-*a.* HC uzima transakcije i generise aktiviranje bus-a delujuci preko paketa sa funkcijama za podatke na datoj strani busa-a za svaku transakciju. Transakcije se ponavljavaju sa svakim USB frejmom ili 1 milisekunde. Ove transakcije mogu da sadrze do tri paketa i to: token, data i handsake i limitirane su velicinom od 8 bajta za spore (low) USB jedinice. Svaka transakcija pocinje od strane HC, iz bazne tabele, slanjem Token paketa jedinicama opisanim tipom i smerom transakcije, USB adresom jedinice i broja krajnje tacke (endpoint number). Adresirana USB jedinica selektuje
ga dekodiranjem odgovarajuceg adresnog polja. Tok podataka moze biti samo u jednom smeru u datom vremenu, bilo od hosta ka USB jedinici (OUT) ili od jedinice ka hostu (IN). Izvor transkacije je npr. host kada salje podatke jedinici (OUT) ili za USB jedinicu (IN). Uredjaj koji prima paket odgovara tzv. Handsake packet-om potvrdjujuci da je transfer zavrsen.

3.7. HID komunikacioni protocol

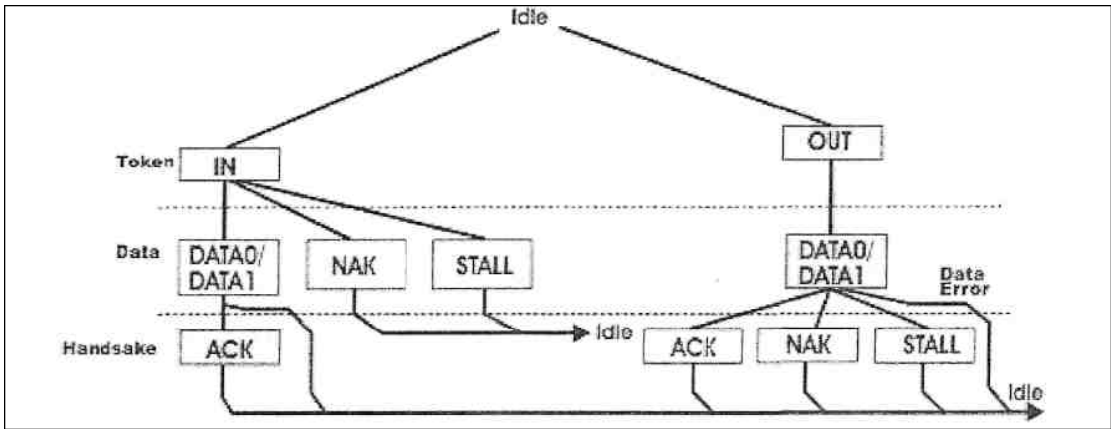
HID jedinice koriste dva od ukupno cetiri USB komunikaciona tipa. Na najnizem nivou, za sve vrste komunikacija izmedju hosta i jedinica koristi se transfer paketa. Na funkcionalnom nivou, komunikacioni tipovi razlikuju se u formatima podataka, njihovoj velicini kao i u pouzdanosti u prenosu.
USB speifikacija 1.1 omogucuje prenos podataka za tzv. *low speed* (male brzine) USB jedinica. HID jedinice podrzavaju samo *Control Interupt* transfer podataka. Control transfer podataka koristi se uz pomocu USB sistemskog softvera pri konfigurisanju jedinica kada se one prvi put prikljucuju. Interapt transfer podataka koristi se kada se radi sa malom broju podataka i ogranicen je na vreme cekanja odziva. Control transfer komunikacija prikazana je na slici 19.



Sl. 19 Format kontrol transakcije

Sl. 19 Format kontrol transakcije

Kako se vidi iz slike *Contol transfer format*, inicira host na taj nacin sto salje SETUP paket u funkciji, primarnoj jedinici za prijem podataka. *Host* zatim salje *Data* paket, koji prilikom prijema se potvrdjuje slanjem ACK paketa prem hostu. Interapt transakcija prikazana je na slici 20.



Sl. 20 Format Interapt transakcije

Iz slike se vidi da ova transakcija pocinje jednim IN ili OUT paketom. Paket signal funkcija moze biti slanje podataka hosta (TN*)* ili prijem podataka hosta (OUT). Jedan poslati ili primljeni podatak hosta, zavrsava se sa ACK paketom ako je podatak primljen ispravno. Ako prijem IN paketa nije izvrsen od strane jedinice, ona salje NAK ili STALL paket hostu*.* Slicno tome ako se pojavi greska u prijemu DATA paketa,

jedinica salje NAK ili STALL paket host-u. Pre nego sto USB jedinica pocne da komunicira sa PC, HID *Interface* mora biti prepoznat kao HID jedinica i detektovana da je prikljucena na USB port. Zbog toga kada se
HIDInterface prvi put prikljuci na USB port, ulazi se u konfiguraciono stanje kada host odreduje fukcije i opseg koji se zahtevaju. Informacije o konfiguraciji sadrzane su obicno u ROM sekciji jedinice i organizovane su pomocu deskriptora.

4. UOPSTENO O MIKROKONTROLERIMA. MIKROKONTROLER PIC 18F2550

4.1 Mikrokontroleri

Mikrokontroler je digitalna elektronska naprava u obliku integrisanog kola. Namena mikrokontrolera je upravljanje uredjajima I procesima, pa u sebi ima integrisan mikroprocesor, memoriju, digitalne I analogne ulaze I izlaze, digitalne satove (timers), brojace (counters), oscilatorem komunikacione sklopove (interface) I druge dodatke za koje je nekada bio potreban niz posebnih integralnih kola (cipova). Mikrokontroler normalno radi u kontrolnoj petlji, dakle ocitava ulaze I zatim podesava izlaze u skladu sa svojim programom. Petlja se stalno ponavlja dok traje kontrola procesa.

Glavna razlika izmedju modernih mikroprocesora I mikrokontrolera je da su prvi optimizovani za brzinu I performance, dok su mikrokontroleri optimizovani u pravcu integracije kola, upravljanja procesima u stvarnom vremenu (real-time control), masovnu proizvodnju, nisku cenu I malu potrosnju struje.

4.1.1 Istorija mikrokontrolera

Sa pojavom prvih mikroprocesora 1971. godine, pocela je I njihova upotreba u kontrolne svrhe. Medjutim tipican sistem je zahtevao veliki broj dodatnih kola za rad, kao sto su bili AD pretvaraci (A/D converters), brojaci, oscilatori I drugo. Vremenom je doslo do integrisanja potrebnih komponenti u jedno kolo I tako je stvoren moderan mikrokontroler.

Jedan od prvih je bio Motorola 6801 mikrokontroler, razvijen od 6800 mikroprocesora. Kasnije, 1985. godine je od 6801 stvoren popularni 68HC11 sa tad novom HCMOS tehnologijom, koja je omogucila manju potrosnju, manju osetljivost na smetnje I brzi rad. Danas mnogi modeli mogu da rade I vrse kontrolu manjih uredjaja bez ikakvih spoljnih delova ili sa minimalnim brojem istih. Proizvodnja mikrokontrolera iznosi nekoliko milijardi godisnje I znatno

premasuje proizvodnju mikroprocesora, tipicnih za licne kompjutere (personal computers – PC).

4.1.2 Delovi mikrokontrolera

Razne vrste ce imati razne module integrisane u mikrokontroler. Ipak vecina ukljucuje kao minimum sledece delove:

- Mikroprocesor
- ROM, EPROM, EEPROM ili fles memoriju (flash memory)
- Oscilator
- Sat (timer)
- Brojac (counter)
- Vocdog tajmer (watchdog timer)
- Digitalne ulaze I izlaze

Vecina njih uz te poseduje I:

- Analogne ulaze I izlaze
- Komunikacioni sklop (interface), USART, SSP I druge
- Analogne poredjivace napona (naponske komparatore)
- Modulator sirine pulsa (PWM modulator) za kontrolu motora

4.1.3 Upotreba

Koriste se u najrazlicitijim modernim uredjajima. U robotima, telekomunikacionim uredjajima, satelitima, automobilima, instrumentima, mobilnim telefonima, kamerama, kucnim uredjajima kao sto su masina za pranje rublja, mikrotalasne rerne, kucnim pekrama hleba I drugde.

4.1.4 Buducnost

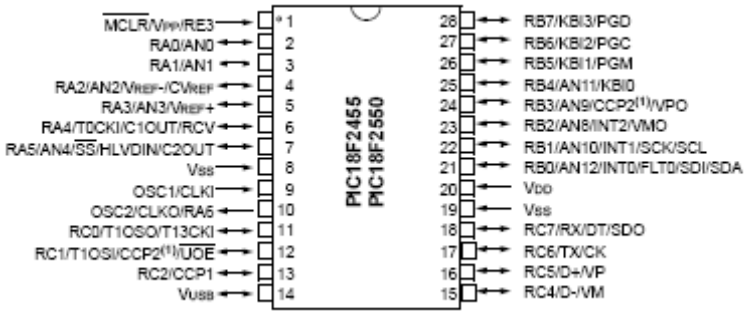
Mikrokontroleri su od sedamdesetih godina proslog veka imali brz razvoj. Sve vise I vise ranije odvojenih kola je integrisano, programiranje je olaksano uvodjenjem flash memoraja (koje se mogu mnogo puta brisati I pisati), smanjena je potrosnja struje (vazno za baterijske uredjaje), a ponuda razlicitih

kontrolera je neverovatno siroka. Danas takodje postoje I 8, 16, 32 I 64 bitni modeli, kao I DSP modeli podeseni za brze matematicke operacije sa prosirenim setom instrukcija.

4.2 Mikrokontroler PIC 18F2550

PIC 18F2550 je 28-pinski (DIP) kontroler koji u sebi ima podrsku za USB komunikaciju.
Izbor odgovarajuceg mikrokontrolera za ovaj projekat je bio uslovljen pre svega na podrsku za usb komunikaciju dok je asinhroni serijski modul implementiran u gotovo svim novijim mikrokontrolerima. Zbog toga je upotrebljen PIC 18F2550 koji po karakteristikama premasuje poptrebe projekta ali svojom niskom cenom i dostupnoscu na trzistu nije imao alternativu.
Pic 18F2550 pripada 18F seriji mikrokontrolera kompanije Microchip. Mikrokontroleri ovog proizvođaca se odlikuju malom cenom i sto je najvaznije najboljom besplatnom tehnickom podrskom (kompajleri, razvojni sistemi programatori). Ovi mikroprocesori imaju *harvard* strukturu pa je memorijska mapa podeljena na programsku i memoriju za podatke kao i EEPROM. CPU koristi tehniku preklapanja kako bi sve instrukcije (osim grananja) izvrsavale jedan ciklus. Zbog toga se osnovni takt deli sa 4 jer se faze izvrsenja naredbi preklapaju. Sve naredbe su fiksne duzine od 2 bajta tako da je adresiranje memorije ograniceno. Zbog toga se memorija deli na 16 stranica a izbor stranice se vrsi u odgovarajucim kontrolnim registrima. Ova osobina znacajno usporava rad mikrokontrolera međutim napredniji kompajleri vrse pametno planiranje raspodele memorije kako bi se varijable koje se zajedno koriste nalazile u istoj memorijskoj banci. Programska memorija je 32KB dok je RAM velicine 2 KB. Takođe postoji i 256B EEPROM-a. Procesor poseduje prosireni skup instukcija u odnosu na ranije serije (16 i 17) kao i nove nacine adresiranja. Tako su dodate naredbe za hardversko mnozenje i deljenje, inkrementiranje i dekrementriranje sa uslovnim skokom, naredbe za citanje tabela i druge.Procesor poseduje I stek ali se on nazaost moze koristiti samo indirektno tako sto se u poseban registar upisuje zeljeni sadrzaj i potom posebnom instrukcijom

sadrzaj tog registra stavlja na stek. Kod citanja sa steka vrednost se takođe nalazi u tom registru. Oscilator pruza brojne mogucnosti prilikom izbora radnog takta koji je ujedno i takt za periferije. Maksimalni eksterni takt je 48MHZ sto daje CPU takt od 12MHZ. Za to se koristi PLL kolo i delitelji frekvencije. Najbitnije je da se za rad usb modula mora obezbediti takt od 24MHZ , a PIC ima prednost u odnosu na konkurenciju sto taj takt moze biti nezavistan od takta za CPU i druge jedinice. Mehanizam interapta je organizovan kao jedan interapt vektor koji sadrzi adresu prekidne rutine u kojoj se treba ispitati izvor prekida i preduzeti zeljena akcija, dakle nema interapt vektora za svaki ili grupu izvora prekida sto je jedan od nedostataka ovog mikrokontrolera jer se time gubi na brzini i preglednosti koda. Ovaj mikrokontroler ima bogat skup hardverskih periferija koje mu omogucavaju primenu u gotovo svim aplikacijama. Mikrokontroler poseduje 20 izlazno ulaznie linije, 4 linije za napajanje, 2 linije za oscilator I po jedna linija za programator i USB kondenzator respektivno, kao sto je prikazano na slici:



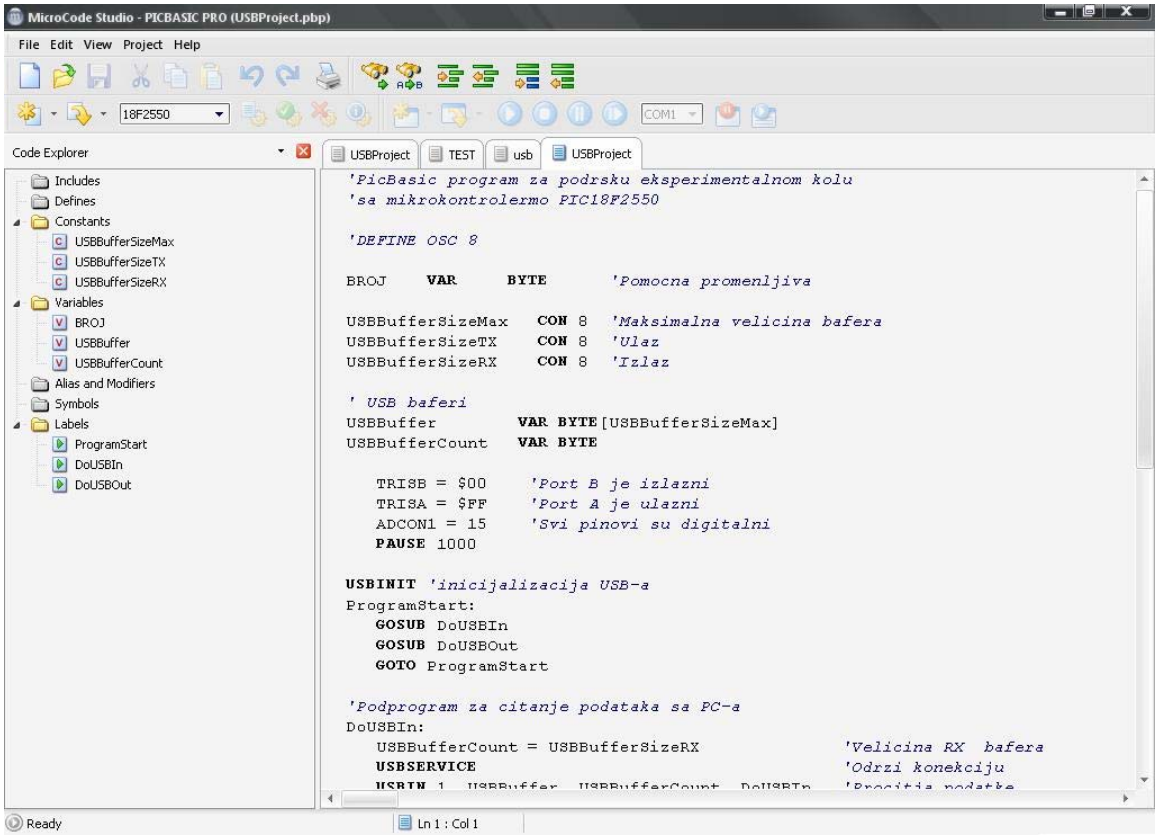
5. KOMPAJLIRANJE I UPISIVANJE PROGRAMA U MIKROKONTROLER

Upisivanje programa u PIC mikrokontroler se vrsi u vise koraka:

1. Pisanje koda programa
2. Generisanje HID modula
3. Kompajliranje I asembliranje
4. Programiranje mikrokontrolera

5.1 Pisanje koda programa

Kod programa koji je dat u sledecoj sestoj oblasti pisan je pomocu programa MicroCode Studio. Izgled samog programa dat je na slici ispod

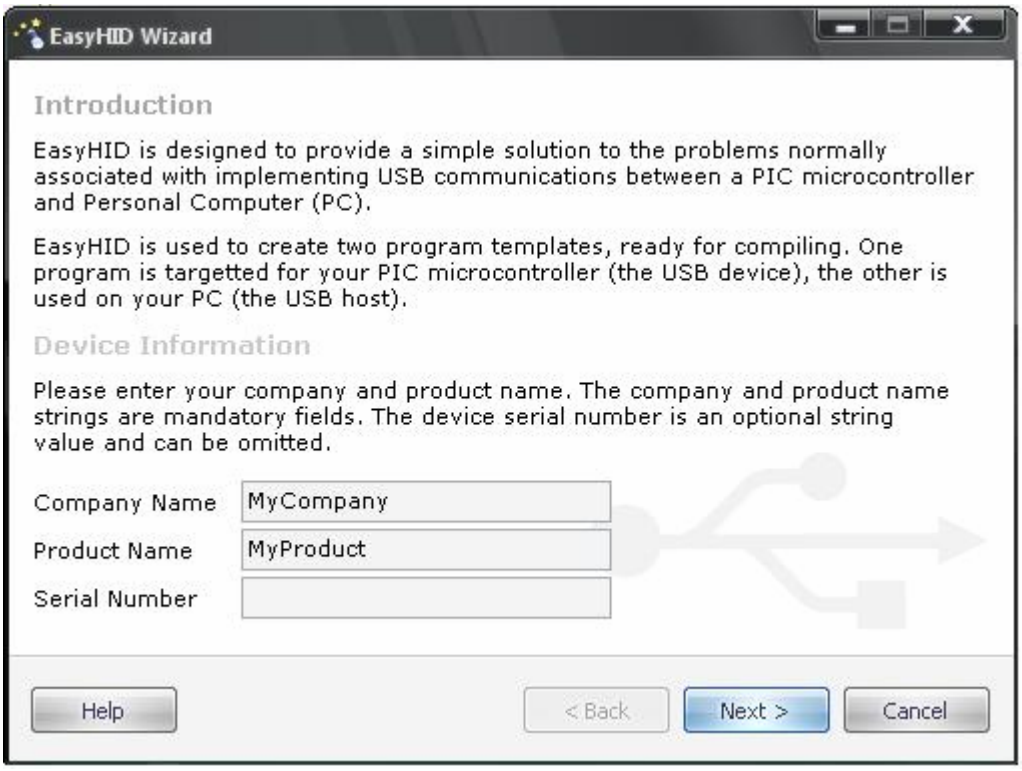


MicroCode Studio je mocan, ali jednostavan program za rukovanje, sa IDE I ICD sposobnoscu dizajniranja posebno za microEngineering Labs PICBASIC ™ I PICBASIC PRO ™ kompajlerom. Podesiti Asemblera I kompajler se lako vrsi

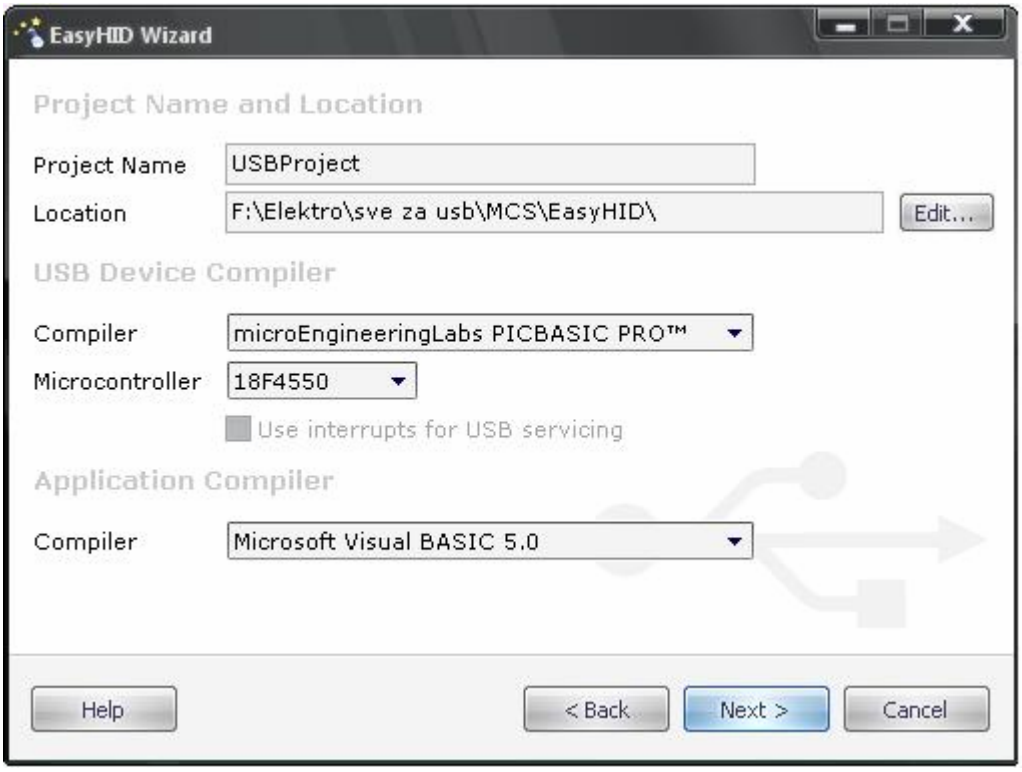
putem autosearch funkcije. Greske u kompajliranju se prikazuju u dnu ekrana I lako se pronalaze.

5.2 Generisanje HID modula

Generisanje HID modula se vrsi preko posebnog wizarda EasyHID a koji se nalazi u sklopu MicroCode studija. Otvaranje EasyHID-a se vrsi klikom na View u meni baru, a zatim klikom na EasyHID USB Wizard. Izgled ovog wizarda dat je na slici ispod



U polje Company Name se upisuje ime proizvodjaca, a u polje Product Name ime proizvoda. Polje Serial Number se ostavlja prazno. Ovaj wizard se sastoji iz par koraka, ali ovde ce biti objasnjen samo onaj najvazniji. Najvazniji korak je odabir kompajlera I mikrokontrolera koji se koriste.



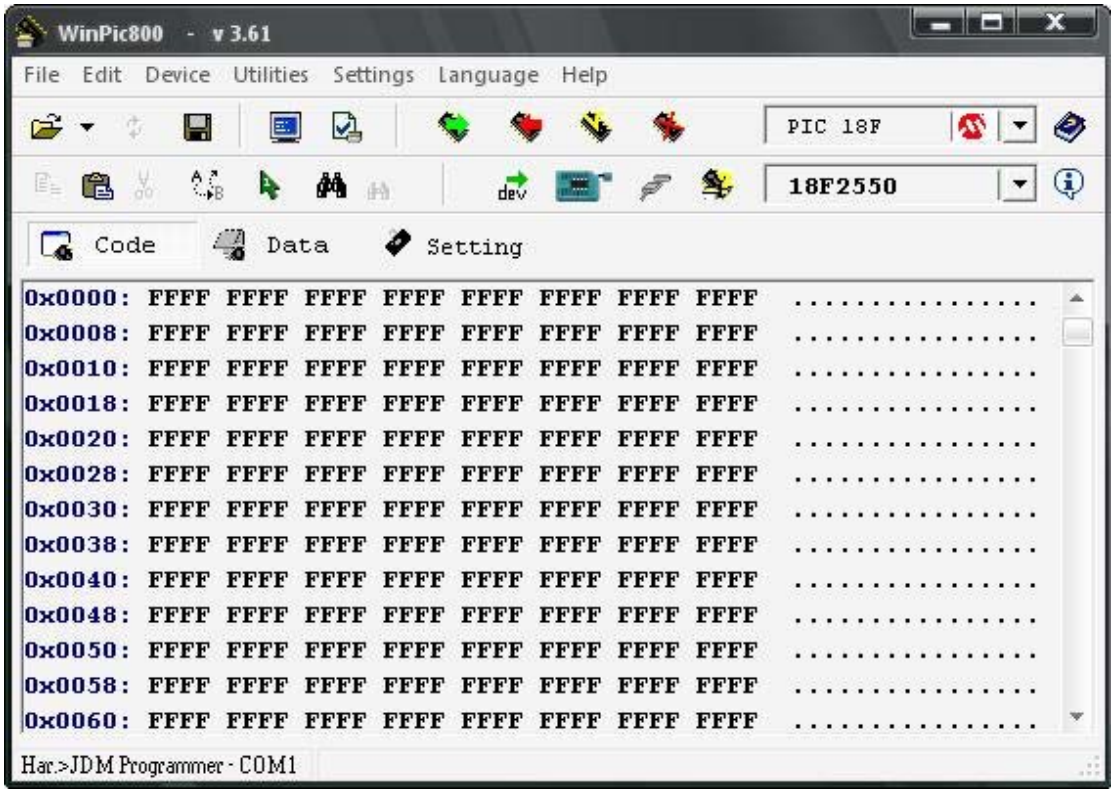
Lokacija za generisanje se uzima da bude sa sto kraca odnosno da u njenoj stazi (path) ima sto manje karaktera. Ovo je vazno zbog samog MicroCode studija koji ne moze da vidi putanje vece od 62 karaktera. Za USB Device Compiler je uzet gore napomenuti PICBASIC PRO™ verzije 2.50. Polje Microcontroller treba promeniti na 18F2550, a za Application Compiler uzet je Microsoft-ov Visual Basic 5.0 (ne podrzava 6.0 iako je ovde program pisan u njemu).

5.3 Kompajliranje I asembliranje

EasyHID wizardom su stvoreni moduli za PICBasic Pro I Visual Studio. Sa modulima su napravljeni I fajlovi za pisanje koda programa u kojima se nalazi sablon odnosno vec napisane funkcije za USB komunikaciju. Ovim je dosta olaksano programiranje mikrokontrolera koji podrzavanju USB komunikaciju. Kompajliranjem se dobija .hex fajl koji se unosi u mikrokontroler.

5.4 Programiranje mikrokontrolera

Dobijeni .hex fajl se otvara u WinPic800 preko kojeg se on unosi u mikrokontroler. Izgled WinPic800 prikazan je dole na slici

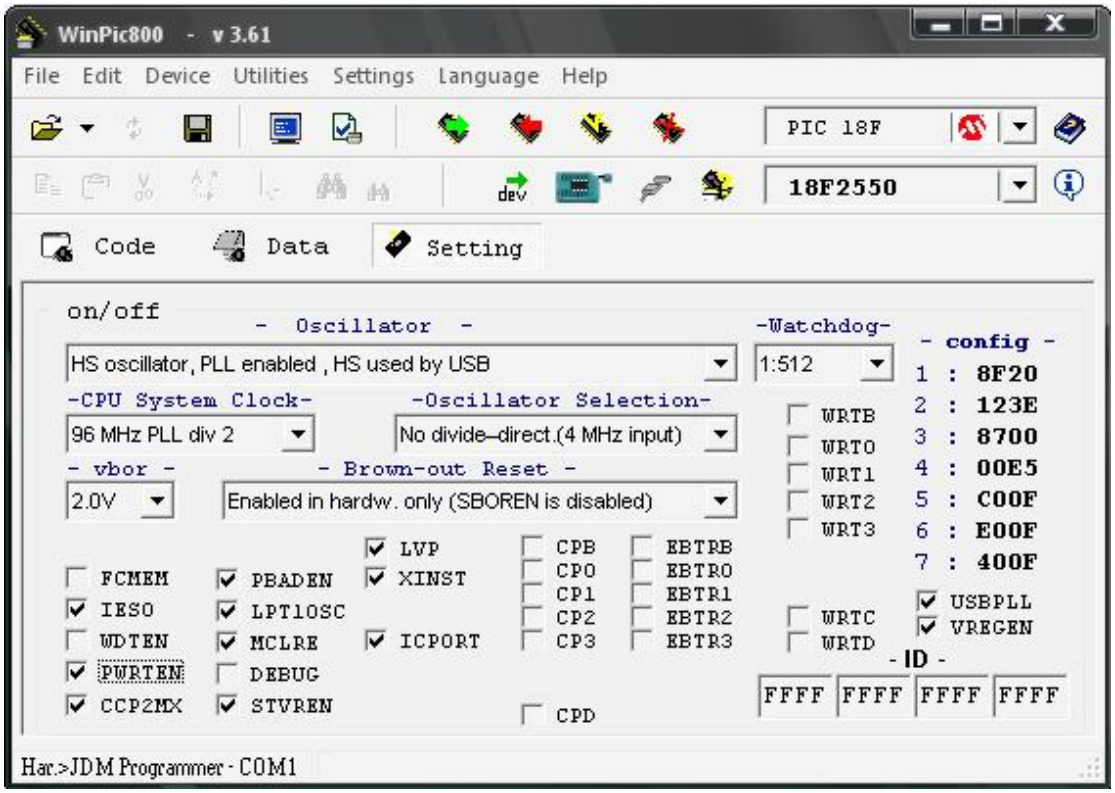


Mikrokontroleri koje podržava WinPIC:

- dsPIC30F2010 PIC16C61, PIC16C71
- PIC16C84, PIC16F84
- PIC16C710, PIC16C711, PIC16C715
- PIC10F200, PIC10F202, PIC10F204, PIC10F206
- PIC12F629, PIC12F635, PIC12F675, PIC12F683
- PIC12F609, PIC16F610, PIC12F615, PIC16F616
- PIC16F627, PIC16F627A, PIC16F628, PIC16F628A
- PIC16F630, PIC16F636, PIC16F648A
- PIC16F676, PIC16F684, PIC16F688
- PIC16F73, PIC16F737, PIC16F74, PIC16F76, PIC16F77
- PIC16F818, PIC16F819
- PIC16F87, PIC16F88
- PIC16F873A...PIC16F877A
- PIC16F88

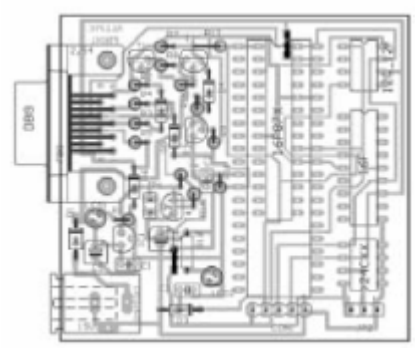
- PIC18F242, PIC18F248, PIC18F252, PIC18F258
- PIC18F442, PIC18F448, PIC18F452, PIC18F458
- PIC18F2XX0/2XX5/4XX0/4XX5 (tested: PIC18F2550)

U opcijama Settings se menjaju kofiguracioni biti bez kojih ovaj projekat ne bi mogao da radi. Za ovaj projekat uzeta su podesavanja kao na slici

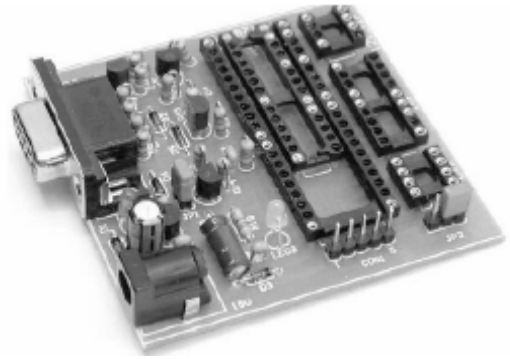


5.4.1 AllPic programator

Da bi se heksadecimalni kod dobijen iz kompilatora/kompajlera (*compiler*) upisao u mikrokontroler, neophodno je posedovati programator. Programator cine dve celine:
bootstrap loader (hardverski deo) i softver (za programiranje firmware-a). Uloga *bootstrap loader*-a je da transformise naponske nivoe porta (DB9, DB25 ili USB) na nivoe standardne logike, dok je uloga softvera da upise heksadecimalni kod u programsku (najcesce: EPROM ili *flash*) memoriju mikrokontrolera. Za potrebe ovog rada korisceni su bootstrap loader ALLPIC i softver WinPic800. Izgled plocice i gotov uređaj – na slikama respektivno



Izgled stampane plocice AllPic



Izgled AllPic-a

6. O KODU PROGRAMA ZA MIKROKONTROLER

Listing ovog programa dat je na kraju ovog poglavlja. Deklaracije kao sto su maksimalna velicina bafera, ulaz, izlaz, usb baferi I podprogrami: inicijalizacija USB, citanje podataka sa PC, upis podataka u PC su generisani pomocu EasyHID wizarda. U listingu su pored svake od naredbi dati I komentari radi lakseg tumacenja programa.

Listing programa:

```
'PicBasic program za podrsku eksperimentalnom kolu
'sa mikrokontrolermo PIC18F2550

BROJ    VAR    BYTE      'Pomocna promenljiva

USBBufferSizeMax   CON 8  'Maksimalna velicina bafera
USBBufferSizeTX    CON 8  'Ulaz
USBBufferSizeRX    CON 8  'Izlaz

' USB baferi
USBBuffer       VAR BYTE[USBBufferSizeMax]
USBBufferCount  VAR BYTE

   TRISB = $00    'Port B je izlazni
   TRISA = $FF    'Port A je ulazni
   ADCON1 = 15    'Svi pinovi su digitalni
   PAUSE 1000

USBINIT 'inicijalizacija USB-a
ProgramStart:
   GOSUB DoUSBIn
   GOSUB DoUSBOut
   GOTO ProgramStart

'Podprogram za citanje podataka sa PC-a
DoUSBIn:
   USBBufferCount = USBBufferSizeRX          'Velicina RX  bafera
   USBSERVICE                                'Odrzi konekciju
   USBIN 1, USBBuffer, USBBufferCount, DoUSBIn   'Procitaj
podatke

   IF USBBuffer[0] = 253 THEN
   Portb = USBBuffer[1]
   ENDIF

   RETURN
```

```
'Podprogram za upis podataka na PC-e
DoUSBOut:
  USBBufferCount = USBBufferSizeTX          'Velicina TX bafera
  USBSERVICE                                'Odrzi konekciju
  USBBuffer[0]=251

  Broj = 0

  IF porta.0 = 1 THEN
  Broj=BROJ + 1
  ENDIF

  IF porta.1 = 1 THEN
  Broj=BROJ + 2
  ENDIF

  IF porta.2 = 1 THEN
  Broj=BROJ + 4
  ENDIF

  USBBuffer[1]=Broj

USBOUT 1, USBBuffer, USBBufferCount, DoUSBOut 'Posalji podatke

RETURN
```

7. O PROGRAMU ZA UPRAVLJANJE EKSPERIMENTALNIM KOLOM

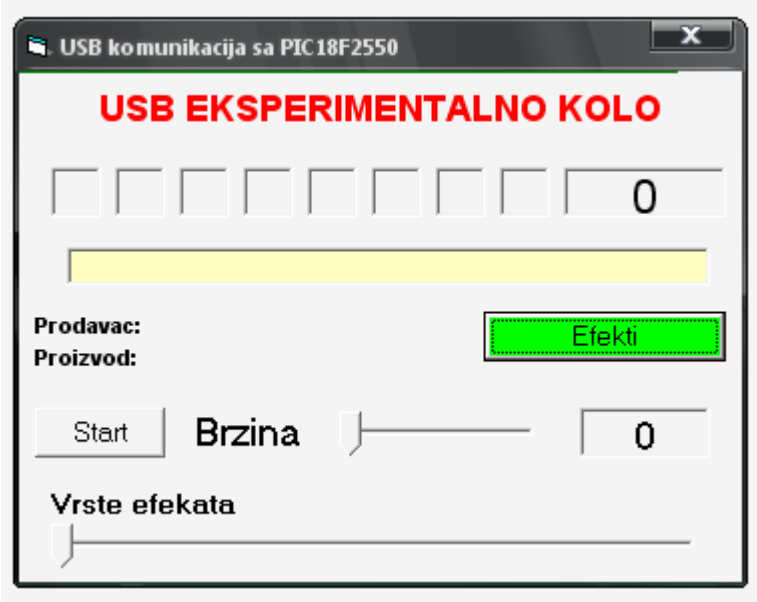
Izgled programa za upravljanje eksperimentalnim kolom dat je na slici

Program se sastoji iz dva dela:

1. Manualno upravljanje LE diodama
2. Upravljanje pomocu efekata

Manualno upravljanje se vrsi klikom na sive kvadratice koji predsatvljaju ugasene LE diode. Kada se klikne na jedan od kvadratica on menja boju a samim tim se ta promena prenosi I na interfejs, odnosno pali se odredjena dioda.

Upravljanje pomocu efekata. Efekti se pale klikom na istoimeno dugme kao sto je prikazano na slici



Pritskom na

dugme start pokrece se jedan od efekata. Efekti se mogu menjati na slideru “Vrste efekata” a njihova brzina na slideru “Brzina”. Brzina je reciprocno izrazena u odnosu na sekunde I moze se menjati od 0 do 100. Efekata ima 8.

Dok su efekti ukljuceni odnosno kada se pritisne dugme Start ono se menja u dugme Stop koje predstavlja dugme za zaustavljanje efekata.

Kada se prikljuci eksperimentalno kolo pored labela Prodavac I Proizvod ispisuju se nazivi koji su uneti u sam mikrokontroler a oni se generisu pomocu EasyHID-a.

Na slici ispod je prikazan program u radu

8. O KODU PROGRAMA ZA UPRAVLJANJE EKSPERIMENTALNIM KOLOM

Ovde ce biti objasnjeno kako je pisan program iz prethodnog poglavlja.
Najveci deo koda zauzimaju efekti koji ovde nisu svi napisani. Kao I za program za mikrokontroler, EasyHID je generisao deo koda I za Visual Basic (to su komande za definisanje bafera, ulaza I izlaza, podprogram za upis podataka, podprogram koji se pokrece pri ukljucenju HID jedinice I dr). Ovo su nazivi koji su korisceni u kodu a koji su drugacije predstavljeni na interfejsu programa:

Command1	dugme Start/Stop
Command2	dugme Efekti
Slider1	Brzina efekata
Slider2	Biranje efekata
Text1	prikaz da li je ukljuceno kolo
Text2	brojni prikaz kombinacije LED
Text3	prikaz brzine efekata
Label1(0)	LE dioda 1 (brojno prikazana kao 1)
Label1(1)	LE dioda 2 (brojno prikazana kao 2)
Label1(2)	LE dioda 3 (brojno prikazana kao 4)
Label1(3)	LE dioda 4 (brojno prikazana kao 8)
Label1(4)	LE dioda 5 (brojno prikazana kao 16)
Label1(5)	LE dioda 6 (brojno prikazana kao 32)
Label1(6)	LE dioda 7 (brojno prikazana kao 64)
Label1(7)	LE dioda 8 (brojno prikazana kao 128)

Srz ovog programa predstavlja podprogram za odredjivanje stanja dioda. Princip rada je sledeci:
Preko *for* petlje se proverava koja od labela je pritisnuta odnosno koja je crvena. Ako je neka od njih crvena ‘RGB (255,0,0)’ onda se promenljiva *Broj* povecava za brojni prikaz (pogledaj gore) labele koja je upaljena. Dalje se ta promenljiva salje u *BufferOut(2),* a iz bafera na interfejs kola.
Pauza koja je koriscena uzeta je iz biblioteke kernel32 I prikazana je u milisekundama.

```
Private Declare Sub sleep Lib "kernel32" Alias "Sleep" (ByVal
dwMilliseconds As Long)
Option Explicit
    '
    ' Konstante za vendor i product
    Private Const VendorID = 6017
    Private Const ProductID = 2000
    '
    '   Definisanje Bafera
    '
    Private Const BufferInSize = 8
    Private Const BufferOutSize = 8
    Dim BufferIn(0 To BufferInSize) As Byte
    Dim BufferOut(0 To BufferOutSize) As Byte
    '   Ostale promenljive
    Public Adresa As Byte
    Public Broj, Broj1, i, j, k As Integer

Private Sub Command1_Click()
If Command1.BackColor = &H8000000F Then
Command1.BackColor = RGB(255, 0, 0): GoTo start
Else
Command1.BackColor = &H8000000F: GoTo kraj
End If
start:

Select Case Slider2
' Prvi Efekat - potrebna dorada sa labelom start:
Case 1
For j = 0 To 7
Pauza
Label1(j).BackColor = RGB(255, 0, 0)
prikazi
Pauza
Label1(j).BackColor = &H8000000F
prikazi
WriteSomeData
Next j
If Command1.BackColor = &H8000000F Then GoTo kraj
GoTo start

' Drugi Efekat - potrebna dorada sa labelom start:
Case 2
For j = 0 To 7
Pauza
Label1(j).BackColor = RGB(255, 0, 0)
prikazi
WriteSomeData
```

```
Next j
For k = 0 To 7
Pauza
Label1(k).BackColor = &H8000000F
prikazi
WriteSomeData
Next k
If Command1.BackColor = &H8000000F Then GoTo kraj
GoTo start

' Treci Efekat - potrebna dorada sa labelom start:
Case 3
Label1(0).BackColor = RGB(255, 0, 0): Label1(7).BackColor =
RGB(255, 0, 0)
prikazi
Pauza
Label1(1).BackColor = RGB(255, 0, 0): Label1(6).BackColor =
RGB(255, 0, 0)
prikazi
Pauza
Label1(2).BackColor = RGB(255, 0, 0): Label1(5).BackColor =
RGB(255, 0, 0)
prikazi
Pauza
Label1(3).BackColor = RGB(255, 0, 0): Label1(4).BackColor =
RGB(255, 0, 0)
prikazi
Pauza
Label1(0).BackColor = &H8000000F: Label1(7).BackColor =
&H8000000F
prikazi
Pauza
Label1(1).BackColor = &H8000000F: Label1(6).BackColor =
&H8000000F
prikazi
Pauza
Label1(2).BackColor = &H8000000F: Label1(5).BackColor =
&H8000000F
prikazi
Pauza
Label1(3).BackColor = &H8000000F: Label1(4).BackColor =
&H8000000F
prikazi
Pauza
WriteSomeData
If Command1.BackColor = &H8000000F Then GoTo kraj
GoTo start
```

```
' Cetvrti Efekat - potrebna dorada sa labelom start:
Case 4
Label1(0).BackColor = RGB(255, 0, 0): Label1(2).BackColor =
RGB(255, 0, 0): Label1(4).BackColor = RGB(255, 0, 0):
Label1(6).BackColor = RGB(255, 0, 0)
prikazi
Pauza
Label1(0).BackColor = &H8000000F: Label1(2).BackColor =
&H8000000F: Label1(4).BackColor = &H8000000F:
Label1(6).BackColor = &H8000000F
prikazi
Label1(1).BackColor = RGB(255, 0, 0): Label1(3).BackColor =
RGB(255, 0, 0): Label1(5).BackColor = RGB(255, 0, 0):
Label1(7).BackColor = RGB(255, 0, 0)
prikazi
Pauza
Label1(1).BackColor = &H8000000F: Label1(3).BackColor =
&H8000000F: Label1(5).BackColor = &H8000000F:
Label1(7).BackColor = &H8000000F
prikazi
Label1(0).BackColor = RGB(255, 0, 0): Label1(2).BackColor =
RGB(255, 0, 0): Label1(4).BackColor = RGB(255, 0, 0):
Label1(6).BackColor = RGB(255, 0, 0)
prikazi
Pauza
Label1(0).BackColor = &H8000000F: Label1(2).BackColor =
&H8000000F: Label1(4).BackColor = &H8000000F:
Label1(6).BackColor = &H8000000F
prikazi
Label1(1).BackColor = RGB(255, 0, 0): Label1(3).BackColor =
RGB(255, 0, 0): Label1(5).BackColor = RGB(255, 0, 0):
Label1(7).BackColor = RGB(255, 0, 0)
prikazi
Pauza
Label1(1).BackColor = &H8000000F: Label1(3).BackColor =
&H8000000F: Label1(5).BackColor = &H8000000F:
Label1(7).BackColor = &H8000000F
prikazi
Label1(0).BackColor = RGB(255, 0, 0): Label1(2).BackColor =
RGB(255, 0, 0): Label1(4).BackColor = RGB(255, 0, 0):
Label1(6).BackColor = RGB(255, 0, 0)
prikazi
Pauza
Label1(0).BackColor = &H8000000F: Label1(2).BackColor =
&H8000000F: Label1(4).BackColor = &H8000000F:
Label1(6).BackColor = &H8000000F
prikazi
```

```
Label1(1).BackColor = RGB(255, 0, 0): Label1(3).BackColor =
RGB(255, 0, 0): Label1(5).BackColor = RGB(255, 0, 0):
Label1(7).BackColor = RGB(255, 0, 0)
prikazi
Pauza
Label1(1).BackColor = &H8000000F: Label1(3).BackColor =
&H8000000F: Label1(5).BackColor = &H8000000F:
Label1(7).BackColor = &H8000000F
prikazi
WriteSomeData
If Command1.BackColor = &H8000000F Then GoTo kraj
GoTo start

' Peti Efekat - potrebna dorada sa labelom start:
Case 5
Label1(0).BackColor = RGB(255, 0, 0)
prikazi
Pauza
Label1(0).BackColor = &H8000000F
prikazi
Pauza
Label1(0).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(1).BackColor = RGB(255, 0, 0)
prikazi
Pauza
Label1(0).BackColor = &H8000000F: prikazi: Pauza:
Label1(1).BackColor = &H8000000F
prikazi
Pauza
Label1(0).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(1).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(2).BackColor = RGB(255, 0, 0)
prikazi
Pauza
Label1(0).BackColor = &H8000000F: prikazi: Pauza:
Label1(1).BackColor = &H8000000F: prikazi: Pauza:
Label1(2).BackColor = &H8000000F
prikazi
Pauza
Label1(0).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(1).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(2).BackColor = RGB(255, 0, 0): Label1(3).BackColor =
RGB(255, 0, 0)
prikazi
Pauza
Label1(0).BackColor = &H8000000F: prikazi: Pauza:
Label1(1).BackColor = &H8000000F: prikazi: Pauza:
```

```
Label1(2).BackColor = &H8000000F: prikazi: Pauza:
Label1(3).BackColor = &H8000000F
prikazi
Pauza
Label1(0).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(1).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(2).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(3).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(4).BackColor = RGB(255, 0, 0)
prikazi
Pauza
Label1(0).BackColor = &H8000000F: prikazi: Pauza:
Label1(1).BackColor = &H8000000F: prikazi: Pauza:
Label1(2).BackColor = &H8000000F: prikazi: Pauza:
Label1(3).BackColor = &H8000000F: prikazi: Pauza:
Label1(4).BackColor = &H8000000F
prikazi
Pauza
Label1(0).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(1).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(2).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(3).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(4).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(5).BackColor = RGB(255, 0, 0)
prikazi
Pauza
Label1(0).BackColor = &H8000000F: prikazi: Pauza:
Label1(1).BackColor = &H8000000F: prikazi: Pauza:
Label1(2).BackColor = &H8000000F: prikazi: Pauza:
Label1(3).BackColor = &H8000000F: prikazi: Pauza:
Label1(4).BackColor = &H8000000F: prikazi: Pauza:
Label1(5).BackColor = &H8000000F
prikazi
Pauza
Label1(0).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(1).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(2).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(3).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(4).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(5).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(6).BackColor = RGB(255, 0, 0)
prikazi
Pauza
Label1(0).BackColor = &H8000000F: prikazi: Pauza:
Label1(1).BackColor = &H8000000F: prikazi: Pauza:
Label1(2).BackColor = &H8000000F: prikazi: Pauza:
Label1(3).BackColor = &H8000000F: prikazi: Pauza:
Label1(4).BackColor = &H8000000F: prikazi: Pauza:
```

```
Label1(5).BackColor = &H8000000F: prikazi: Pauza:
Label1(6).BackColor = &H8000000F
prikazi
Pauza
Label1(0).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(1).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(2).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(3).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(4).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(5).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(6).BackColor = RGB(255, 0, 0): prikazi: Pauza:
Label1(7).BackColor = RGB(255, 0, 0)
prikazi
Pauza
Label1(0).BackColor = &H8000000F: prikazi: Pauza:
Label1(1).BackColor = &H8000000F: prikazi: Pauza:
Label1(2).BackColor = &H8000000F: prikazi: Pauza:
Label1(3).BackColor = &H8000000F: prikazi: Pauza:
Label1(4).BackColor = &H8000000F: prikazi: Pauza:
Label1(5).BackColor = &H8000000F: prikazi: Pauza:
Label1(6).BackColor = &H8000000F: prikazi: Pauza:
Label1(7).BackColor = &H8000000F
prikazi
If Command1.BackColor = &H8000000F Then GoTo kraj
GoTo start

WriteSomeData

If Command1.BackColor = &H8000000F Then GoTo kraj
GoTo start
End Select
kraj:

End Sub

Private Sub Command2_Click()
If Command2.BackColor = &H8000000F Then
MainForm.Height = 4275
Else
MainForm.Height = 2775
End If

If Command2.BackColor = &H8000000F Then
Command2.BackColor = RGB(0, 255, 0): Command1.Visible =
True: Slider1.Visible = True: Text3.Visible = True: Slider2.Visible =
True: Label3.Visible = True: Label4.Visible = True
Else
```

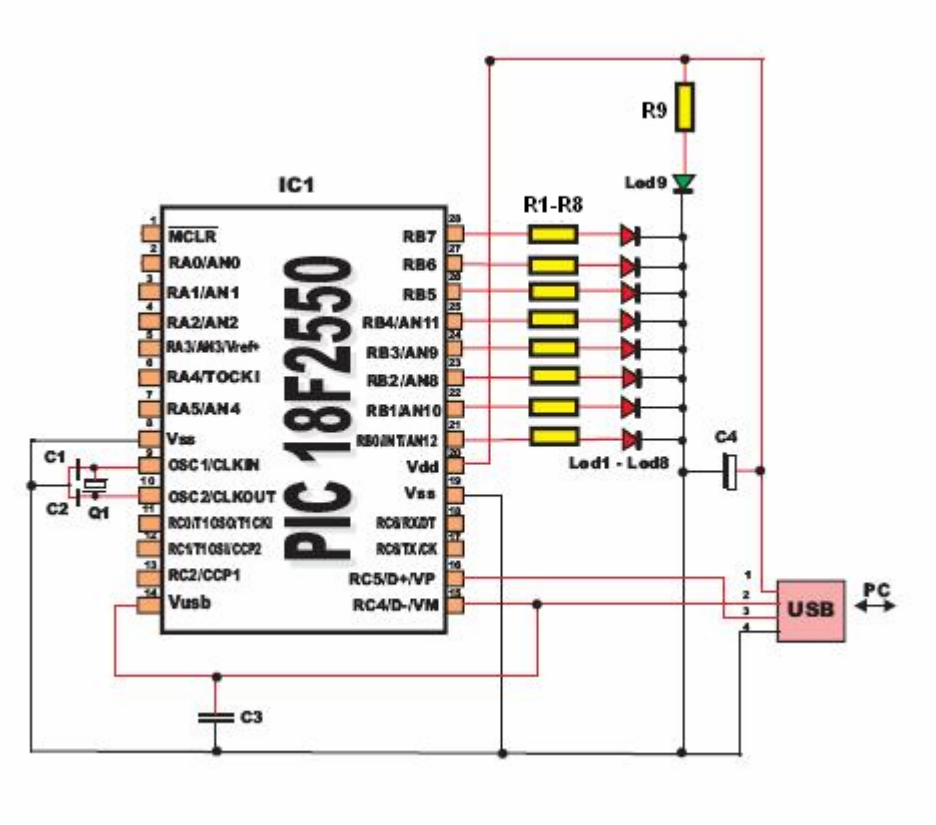
```
Command2.BackColor = &H8000000F: Command1.Visible = False:
Slider1.Visible = False: Text3.Visible = False: Slider2.Visible =
False: Label3.Visible = False: Label4.Visible = False
End If
End Sub

'Podprogram koji se pokrece ucitavanjem forme
Private Sub Form_Load()

   ConnectToHID (Me.hwnd)      'HID konekcija
   lVendorName.Caption = ""
   lProductName.Caption = ""
   Adresa = 253


End Sub
'Podprogram koji se pokrece zatvaranjem forme
Private Sub Form_Unload(Cancel As Integer)
  DisconnectFromHID
End Sub
'Podprogram koji se pokrece prikljucenje HID jedinice
Public Sub OnPlugged(ByVal pHandle As Long)

   Dim DeviceHandle As Long
   Dim VendorName As String * 15
   Dim ProductName As String * 25

   If hidGetVendorID(pHandle) = VendorID And
hidGetProductID(pHandle) = ProductID Then ' Good one?
      '
      '  preuzmi device handle
      DeviceHandle = hidGetHandle(VendorID, ProductID)
      '
      '  Preuzmi trgovca i proizvod iz Devicehandle
      hidGetVendorName DeviceHandle, VendorName, 255
      hidGetProductName DeviceHandle, ProductName, 255
      '  Prikazi trgovca i proizvod
      lVendorName.Caption = VendorName
      lProductName.Caption = ProductName
      End If

  If hidGetVendorID(pHandle) = VendorID And
hidGetProductID(pHandle) = ProductID Then
    Text1.Text = "Prikljuceno eksperimentalno kolo"
  End If
End Sub

'Podprogram koji se pokrece iskljucenjem HID jedinice
```

```
Public Sub OnUnplugged(ByVal pHandle As Long)
   If hidGetVendorID(pHandle) = VendorID And
hidGetProductID(pHandle) = ProductID Then
      Text1.Text = "Iskljuceno eksperimentalno kolo"
   End If
End Sub
'Podprogram koji se pokrece nakon promene kontrolera
Public Sub OnChanged()
   Dim DeviceHandle As Long
   DeviceHandle = hidGetHandle(VendorID, ProductID)
   hidSetReadNotify DeviceHandle, True
End Sub


'Podprogram za upis podataka u USB kolo
Public Sub WriteSomeData()
   BufferOut(0) = 0      'Prvi bajt je report ID
   BufferOut(1) = Adresa 'Drugi bajt je zahtev USB kolu
   BufferOut(2) = Broj   'Treci bajt je vrednost led dioda
   BufferOut(3) = 0      '0
   BufferOut(4) = 0      '0
   BufferOut(5) = 0      '0
   BufferOut(6) = 0      '0
   BufferOut(7) = 0      '0
   BufferOut(8) = 0      '0

   hidWriteEx VendorID, ProductID, BufferOut(0) 'upisi podatke u
USB
End Sub

'Podprogram za odredjivanje stanja led didoda
Private Sub prikazi()
Broj = 0
For i = 0 To 7
If Label1(i).BackColor = RGB(255, 0, 0) Then Broj = Broj + 2 ^ i
Next i
Text2.Text = Broj
WriteSomeData
End Sub

Private Sub Label1_Click(Index As Integer)


If Label1(Index).BackColor = &H8000000F Then
Label1(Index).BackColor = RGB(255, 0, 0)
Else
Label1(Index).BackColor = &H8000000F
End If
```

```
prikazi
End Sub
'Podprogram koji se pokrece za citanje
Public Sub OnRead(ByVal pHandle As Long)

   ' Provera da je prvi clan nula...
   If hidRead(pHandle, BufferIn(0)) Then
      'Ako jeste primljeni su podaci od USB-a
      'Prvi bajt je report ID, t.j. BufferIn(0)
      'Ostali bajti su bajtovi podataka

       Broj1 = BufferIn(2)

      'prikazi rezultat



   End If
End Sub
Private Sub Pauza()
DoEvents
sleep ((101 - Slider1) * 3)

End Sub


Private Sub Slider1_Click()
Text3.Text = Slider1
End Sub

Private Sub kraj()
Broj = 0

End Sub
```

9. INTEFEJS EKSPERIMENTALNOG KOLA

Interfejs eksperimentalnog kola, kao I njegova sema prikazani su na slici:



Kao sto se I vidi sa slike ovaj interfejs se sastoji od:

1. PIC mikrokontrolera 18F2550
2. USB prikljucka na ulazu
3. LE dioda na izlazu

Za izlazni port uzet je port B (od RB0 do RB7). Otpornici od R1 do R8 sluze za ogranicavanje napona na izlazu mikrokontrolera. Q1 sa kondenzatorima C1 I C2 predstavlja kvarcni rezonator koji u ovom slucaju osciluje frekvencijom od 4MHz. Dioda Led 9 se pali pri prikljucivanju interfejsa na racunar odnosno ona signalizira napon od 5V na USB portu. Kondenzator C3 sluzi da racunaru predstavi ovo kolo kao HID (Human Interface Device).
Za vezu sa racunarom se moze uzeti bilo koji tip USB konektora (A, B, mini A, mini B). U ovoj varijanti je to uradjeno bez konektora.

Literatura

Koriscena literatura preuzeta je sa internet stranica:
www.google.com
sr.wikipedia.com
www.accesscomms.com.au/reference/usb.htm
milan.milanovic.org/skola

Maturski rad Predraga Miletica “Skolsko zvono na serijskom portu”
Projekat Zuhdina Curica “USB komunikacija sa PIC18F4550”

Knjiga “PC Interfejsi” Voje Milanovica