SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 177

Analizator spektra na ugradbenom računalu

Ivan Dodig

Zagreb, lipanj 2008

IZVORNIK ZAVRŠNOG RADA

- mentorom dogovorit kako ga ubaciti

Želim se zahvaliti mentoru **dr. sc. Davoru Petrinović,** jer mi je omogućio izradu ovog završnog rada te pružio svoju pomoć kad god mi je bila potrebna.

Također se zahvaljujem **dr. sc. Hrvoju Džapi,** jer mi je uz puno povjerenje omogućio korištenje razvojnog sustava te pružio korisne materijale vezane uz tematiku.

Sadržaj

Uvod

Predmet radnje ovog završnog rada je izrada analizatora spektra audio signala.

Analizator spektra je uređaj koji ulazni signal iz vremenske domene pretvara u frekvencijsku domenu. U ovome slučaju ulazni signal je realan, analogan audio signal. Kao izvor audio signala korišten je običan prijenosni MP3 *player.*

Spektar signala izračunava ugradbeno računalo, tj. mikrokontroler. Mikrokontroler mora biti dovoljno brz budući da je izračun spektra vremenski zahtjevna operacija. Iz tog razloga odabran je kontroler s ARM7TDMI procesorom.

Audio signal se preko posebnog ulaznog sklopa dovodi na AD pretvornik. Ulazni sklop namješta statičku radnu točku i dinamiku rada.

Odabrani mikrokontroler opisan je u nastavku. On sadrži potreban AD pretvornik rezolucije 10-bita koji služi za uzorkovanje audio signala.

Nakon što mikrokontroler izračuna spektar, on se vizualno atraktivno prikazuje na *Dot Matrix Display-u* veličine 20 x 20 LED dioda.

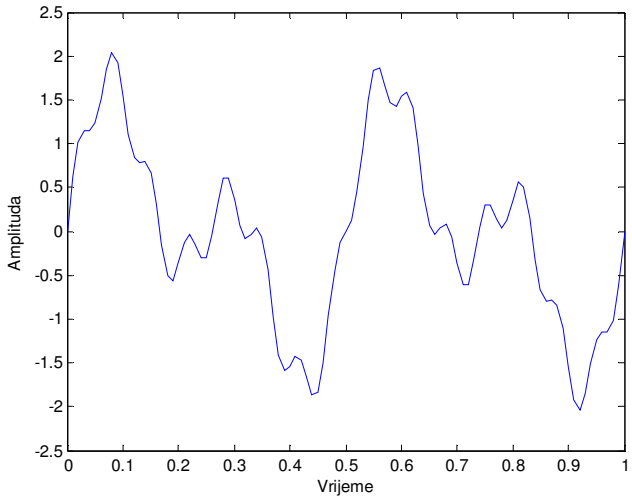
Display je posebno izrađen za ovaj projekt. Njegovo srce opet čini mikrokontroler, samo puno slabijih proporcija. Način rada, hardver i softver *display-a* detaljno su objašnjeni u nastavku.

Osim tehničkog opisa izrade dva navedena sustava, ovaj rad sadrži i teoretsku podlogu koja se krije iza principa na kojima se analizator spektra zasniva. Tu prvotno mislim na Fourierovu transformaciju i njeno fizikalno značenje.

Spektar signala

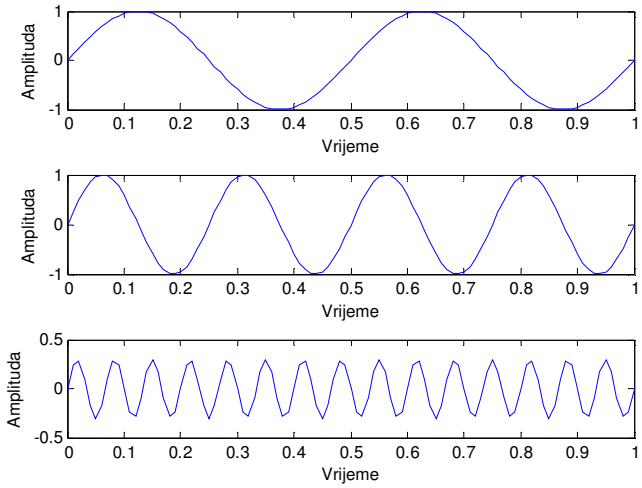
2.1 Uvod

Iako se na prvu ruku čini da je prirodnije promatrati signale u vremenskoj domeni, lako je uvjeriti se da to nije tako. Kao dokaz pogledajmo sljedeći signal.



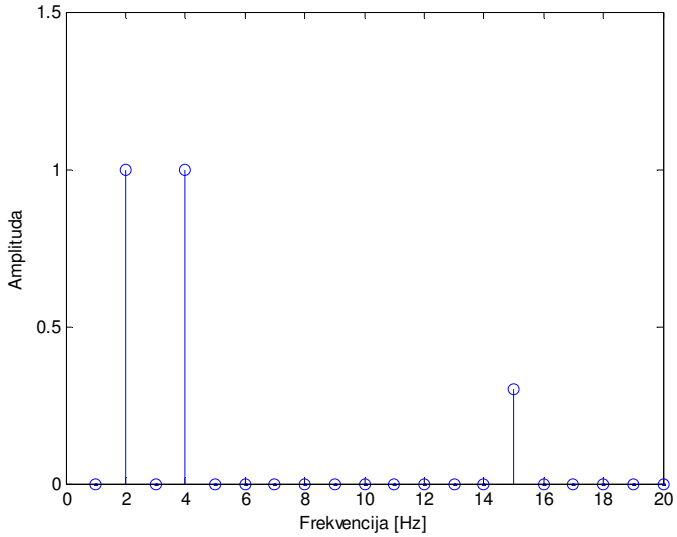
Slika 1. Realan signal

Kada bi vas netko upitao da mu opišete signal na slici, što biste mu rekli? Vjerojatno ne puno, osim da je signal periodičan. Signal na slici predstavlja zbroj različitih sinusnih signala prikazanih u nastavku.



Slika 2. Tri sinusna signala

Možemo vidjeti da se signal s prve slike sastoji od tri sinusa frekvencija redom 2, 4 i 15Hz te amplituda 1, 1 i 0.3. Kombinacija ta dva podatka čini spektar signala. Spektar signala jednoznačno opisuje signal.



Slika 3. Prikaz spektra signala

Umjesto crtanja svih signala koji su *ugrađeni* u promatrani signal, slika tri predstavlja pojednostavljeni prikaz spektra. Šiljak na frekvenciji 2Hz je visok 1. To znači da u spektru signala postoji sinus dotične frekvencije i amplitude. Isto vrijedi i za ostale šiljke.

Danas se u svim granama tehnike signali najčešće obrađuju i prikazuju u frekvencijskom području. Razlog tomu je što se na taj način vrlo složeni signali mogu promatrati i obrađivati pojednostavljeno.

U elektrotehnici je promatranje signala kroz njegov spektar od iznimne važnosti. Kao primjer navodim električni filtar. Njegova svojstva određuje razlika spektra na ulazu i izlazu.

Matematički aparat koji prevodi neki slučajni signal iz vremenske u frekvencijsku domenu, tj. koji prikaz signala u odnosu na vrijeme prevodi u prikaz amplituda harmonika u odnosu na frekvenciju, naziva se ***Fourierova transformacija.***

Fourierova transformacija jedna je od najvažnijih matematičkih transformacija.

2.2 Fourierova transformacija

Jean Baptiste Joseph Fourier (21.3.1768 – 16.05.1830) bio je francuski matematičar fizičar i povjesničar [1].

Povijest ga je najviše zapamtila po uvođenju *Fourierovog reda.* Radeći na problemu širenja topline u čvrstim tijelima, *Fourier* je predstavio javnosti red koji je izveo da bi si olakšao proračune.

Fourierov red rastavlja periodičnu funkciju u beskonačan niz sinusnih harmonika aproksimirajući na taj način funkciju. Pri tome je često u samo prvih par harmonika sadržana većina energije signala.

Iz matematičkih temelja njegovog reda nastala je transformacija nazvana u njegovo ime.

Slika 3. Fourier osobno

Fourierova transformacija omogućava izračun spektra nekog *proizvoljnog* neperiodičnog signala. Budući da je audio signal primjer neperiodičnog signala, Fourierovom transformacijom bit će pretvoren u frekvencijsko područje.

Izraz za Fourierovu transformaciju prikazan je u nastavku

$$X(\omega) = \int_{t=-\infty}^{\infty} f(t) e^{-j\omega t} dt$$

2.3 Fizikalno značenje Fourierove transformacije

2.3.1 Fourierov red

Dobrog inženjera elektrotehnike bi trebala zanimati fizikalna narav Fourierove transformacije, tj. način na koji se do nje dođe. Upravo je to opisano u nastavku teksta.

Kao što je već spomenuto, razumijevanje Fourierovog reda prvi korak je u razumijevanju same transformacije pa objašnjenje kreće od toga.

Fourier je otkrio da se bilo koja obavezno periodična funkcija koja zadovoljava Dirichletove uvjete [2] može rastaviti u beskonačan niz sinusnih signala čije su frekvencije cjelobrojni višekratnici od temeljne frekvencije periodičnog signala.

Sinusni signal je vrlo jednostavan jer mu se lako odredi frekvencija, amplituda i faza. Drugim riječima, Fourierov red rastavlja složen signal u niz jednostavnih funkcija poznatih svojstva.

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t))$$

Navedeni izraz nema neki posebni izvod. Na njega se može gledati empirijski. Sam po sebi bi trebao biti vrlo jasan, jer predstavlja samo matematički zapis Fourierove ideje o rastavljanju periodične funkcije na niz harmonika.

Postavlja se pitanje kako odrediti koeficijente a_0, a_n i b_n.

Intuitivno zaključujemo da prvi član u izrazu predstavlja harmonik nulte frekvencije, tj. neku srednju (DC) vrijednost signala oko koje ostali harmonici tvore signal.

Da bi odredili ostale koeficijente, potrebno je prvo proučiti svojstvo ortogonalnosti trigonometrijskih funkcija.

Za trigonometrijske funkcije cos(t), sin(t), cos(2t), sin(2t)… na intervalu od $-\pi$ do π vrijedi :

$$\int_{-\pi}^{\pi} \cos(nt)dt = \begin{matrix} 2\pi, n = 0 \\ 0, n > 0 \end{matrix}$$

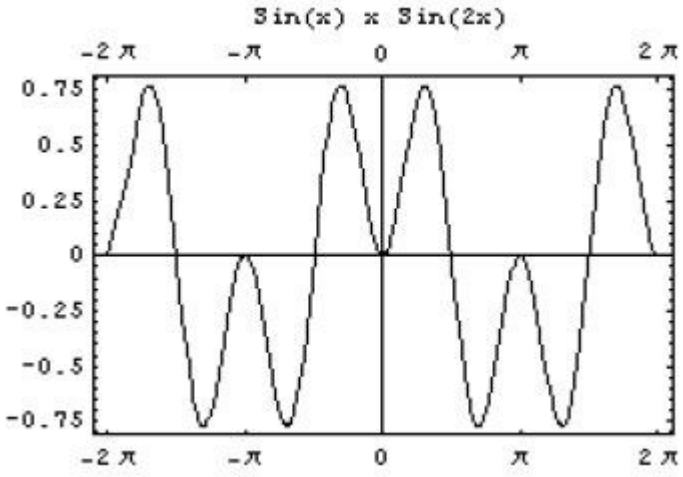
$$\int_{-\pi}^{\pi} \sin(nt)dt = 0$$

$$\int_{-\pi}^{\pi} \sin(nt)\sin(mt)dt = \begin{matrix} 0, m \neq n \\ \pi, m = n \end{matrix}$$

$$\int_{-\pi}^{\pi} \cos(nt)\cos(mt)dt = \begin{matrix} 0, m \neq n \\ \pi, m = n \end{matrix}$$

$$\int_{-\pi}^{\pi} \sin(nt)\cos(mt)dt = 0$$

Do gornjih izraza lako se može doći integriranjem, no pogled na sljedeću sliku dodatno objašnjava svojstvo ortogonalnosti.



Slika 4. Umnožak sinusa – preuzeto iz [3]

Slika jasno pokazuje da je površina ispod krivulje umnoška u promatranom intervalu jednaka nuli.

Odredimo sada koeficijent a_n, tako da pomnožimo izraz za Fourierov red sa cos(mt) te nakon toga integriramo.

$$\int_{-\pi}^{\pi} f(t)\cos(mt)dt = \frac{a_0}{2}\int_{-\pi}^{\pi} \cos(nt)dt + \sum_{n=1}^{\infty}(a_n \int_{-\pi}^{\pi} \cos(nt)\cos(mt)dt + b_n \int_{-\pi}^{\pi} sin(nt)\cos(mt))dt$$

Odmah je vidljivo da su prvi i zadnji član s desne strane jednaki nuli. Srednji član će također uvijek biti jednak nuli, osim u slučaju kad je n=m. U tom slučaju je jednak π. Dakle cijeli desni izraz je jednak upravo $a_n \pi$.

Na sličan način se dobije b_n, množenjem sa sinusom. U nastavku su prikazani konačni izrazi za koeficijente Fourierovog reda.

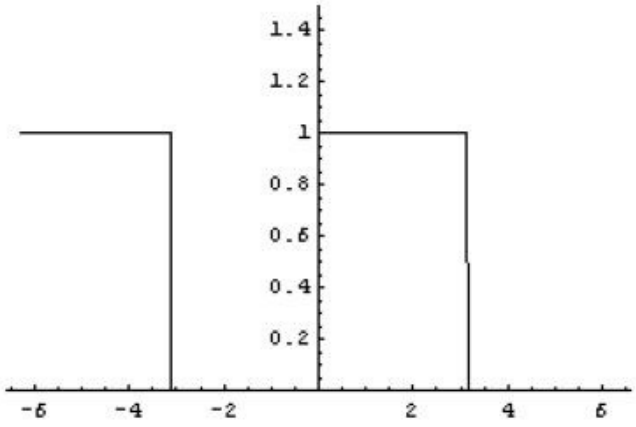
$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)dt$$

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\cos(nt)dt$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\sin(nt)dt$$

2.3.2 Primjer aproksimacije Fourierovim redom

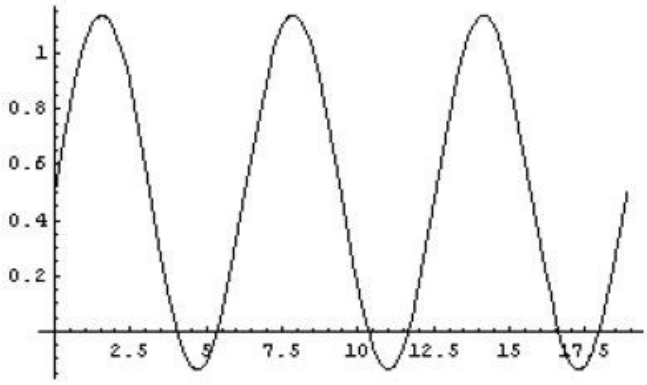
Pogledajmo kako izgleda rastav pravokutnog signala u red.



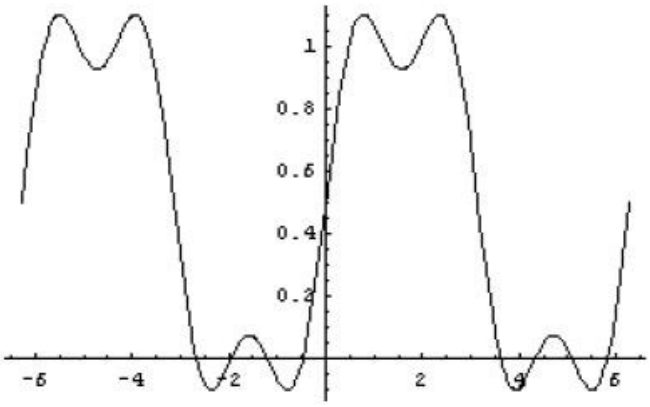
Slika 5. Periodičan pravokutan signal – preuzeto iz [3]

Koristeći gornje izraze za koeficijente reda lako dolazimo do reda za signal na slici.

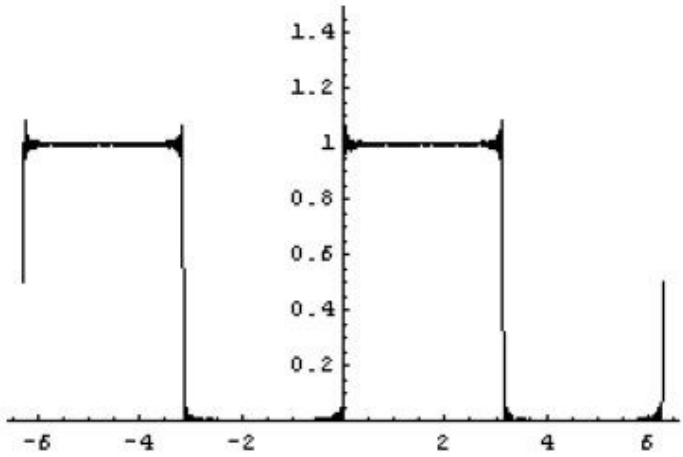
$$f(t) = \frac{1}{2} + \frac{2}{\pi}(\sin(t) + \frac{\sin(3t)}{3} + \frac{\sin(5t)}{5} + ...)$$



Slika 6. DC i prvi harmonik – preuzeto iz [3]



Slika 7. DC i prva dva harmonika – preuzeto iz [3]



Slika 8. Prvih 50 harmonika – preuzeto iz [3]

Očito je da Fourierov red odlično aproksimira pravokutni signal. Smetnja na rubovima bridova je posljedica pokušaja opisivanja diskontinuiteta s glatkim sinusnim funkcijama.

Dobiveni Fourierov red opisuje spektar signala koji rastavljamo. Spektralne komponente su diskretne i udaljene jedna od druge za recipročnu vrijednost osnovnog perioda (u ovom primjeru je amplituda svih parnih harmonika jednaka nuli).

2.3.3 Kompleksni oblik Fourierovog reda

Često se Fourierov red zapisuje u kompleksnom obliku. Razlog tome je što je za razmatranje samog reda kompleksni oblik praktičniji.

Uvrstimo li u

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t))$$

kompleksni zapis sinusa i kosinusa

$$\cos(nt) = \frac{e^{jnt} + e^{-jnt}}{2};\ \sin(nt) = \frac{e^{jnt} - e^{-jnt}}{2j}$$

dobivamo

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} ((\frac{a_n - jb_n}{2})e^{jnt} + (\frac{a_n + jb_n}{2})e^{-jnt}).$$

Prozovemo koeficijente ovakvog reda s

$$c_0 = \frac{a_0}{2};\ \ c_n = \frac{a_n - jb_n}{2}$$

što nakon uvrštenja iznosi

$$f(t) = c_0 + \sum_{n=1}^{\infty} (c_n e^{jnt} + c_{-n} e^{-jnt})$$

Kada sve svedemo pod istu sumu, Fourierov red iznosi

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{jnt}$$

Koeficijent c_n predstavlja *jačinu* pojedinih kompleksnih eksponencijala u signalu $f(t)$. On se jednostavno odredi iz svoje definicije, uvrštavajući u izraz dobivene formule umjesto koeficijenata a_n i b_n.

$$c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(t) e^{-jnt} dt$$

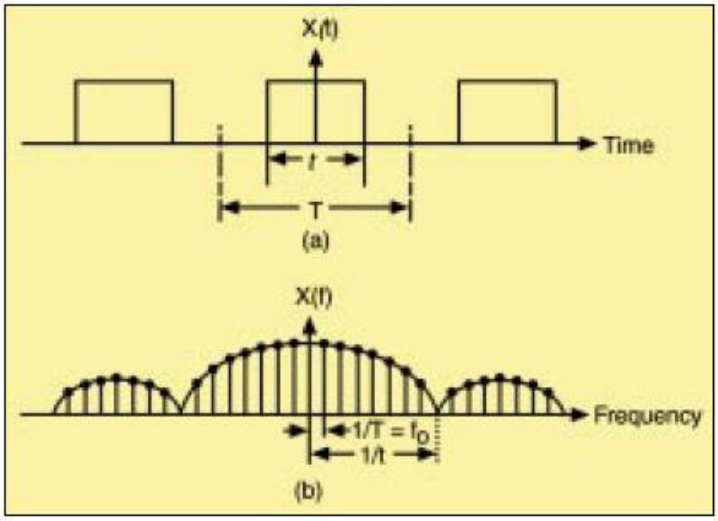
Dobiveni izraz određuje jačinu (i fazu) diskretnih spektralnih komponenata signala $f(t)$. Može se primijetiti da je njegov oblik vrlo sličan samoj Fourierovoj transformaciji.

2.3.4 Kontinuirana Fourierova transformacija

Do sada je pokazano kako se može izračunati spektar periodičnih signala koristeći Fourierov red. Nažalost, Fourierov red nam ne može puno pomoći u izračunu spektra neperiodičnih signala. Većina realnih signala je upravo neperiodična. Primjer je zvuk, slika, video itd.

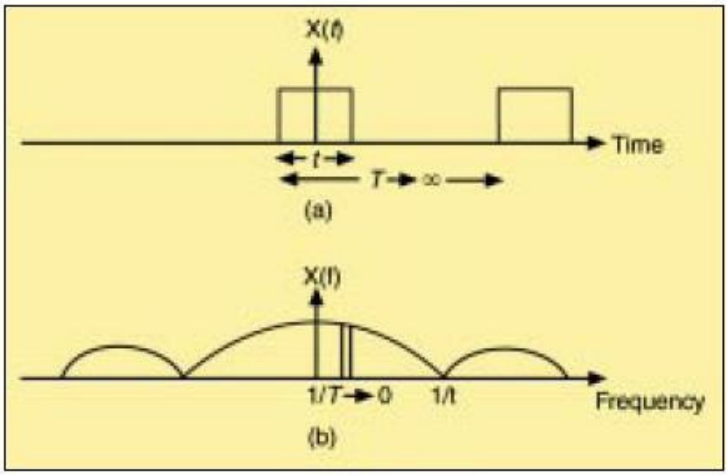
Jednom pametnom prilagodbom Fourierovog reda dolazi se do izraza koji računa spektar neperiodičnih signala. Upravo je to glavna bit Fourierove transformacije.

Promotrimo sada kako izgleda spektar periodičnog niza pravokutnih impulsa izračunat koristeći Fourierov red.



Slika 9. Spektar niza impulsa – preuzeto iz [4]

Kao što je već spomenuto u prošlom poglavlju spektar je diskretan. Spektralne komponente razmaknute su za recipročnu vrijednost temeljnog perioda signala.



Slika 10. Spektar jednog impulsa- preuzeto iz [4]

Ako bismo povećavali period pravokutnog signala, spektralne komponente postajale bi sve gušće i gušće. U krajnjem slučaju, dok period teži u beskonačnost spektar postaje kontinuiran.

Pogledajmo kako se to odražava na formulu za Fourierov red.

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t))$$

Koeficijent a_0 nestaje jer je srednja vrijednost impulsa promatrana kroz beskonačnost nule. Sumator koji zbraja spektralne komponente se zamjenjuje integralom jer spektar više nije diskretan, već kontinuiran. Dakle, diskretne frekvencije $n\omega_0$ sada prelaze u kontinuiranu frekvenciju ω.

$$f(t) = \int_{\omega=0}^{\infty} (a_n \cos(\omega t) + b_n \sin(\omega t)) d\omega$$

Ukoliko gornji izlaz zapišemo u kompleksnom obliku (slično kao u poglavlju 2.3.3) dobivamo

$$f(t) = \int_{-\infty}^{\infty} X(\omega) e^{j\omega t} d\omega$$

Koeficijent $X(\omega)$ predstavlja kontinuiranu jačinu harmonika kompleksne eksponencijale. Drugim riječima, predstavlja spektar signala.

$$X(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$$

Ukratko rezimirano, spektar periodičnih signala je diskretan i može se izračunati koristeći Fourierov red. S druge strane, spektar neperiodičnih signala je kontinuiran i računa se uporabom Fourierove transformacije.

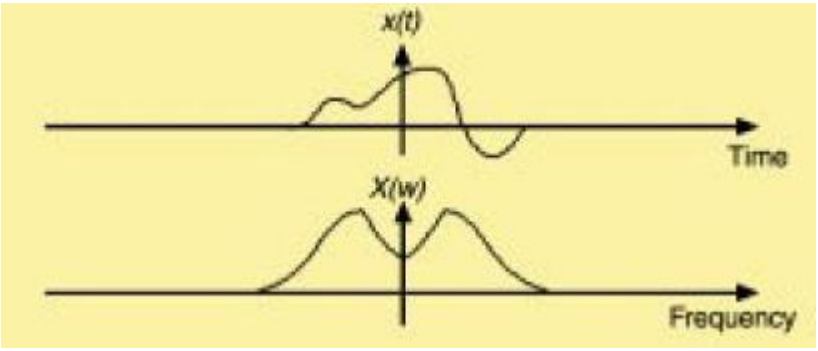
2.3.5 Digitalna računala i Fourierova transformacija

Fourierova transformacija je matematička transformacija signala iz vremenskog područja u frekvencijsko i obratno. Digitalna računala, tj. procesori su diskretni uređaji. Oni nisu sposobni raditi s beskonačnom količinom podataka koje bi unijela jedna kontinuirana funkcija u memoriju računala.

Iz tog je razloga potrebno osigurati metodu izračuna spektra nad diskretnim uzorcima signala koje računalo bez problema obrađuje. Drugim riječima, cilj je formulu za Fourierovu transformaciju prilagoditi digitalnom računalu.

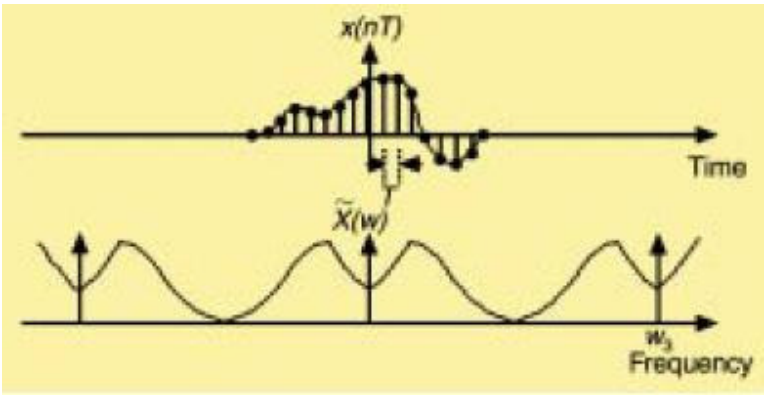
Da bi računalo bilo u mogućnosti izvoditi Fourierovu transformaciju potrebno je

- osigurati ulazni signal konačne duljine i konačne širine spektra (tj. konačne maksimalne frekvencije)
- signal mora biti otipkan konačnim brojem uzoraka
- dobiveni spektar mora imati konačan broj frekvencijskih komponenti



Slika 11. Spektar neperiodičnog kontinuiranog signala – preuzeto iz [4]

Slika 11 prikazuje spektar neperiodičnog kontinuiranog signala. Spektar je također neperiodičan i kontinuiran. Otipkajmo sada signal nekom fiksnom frekvencijom otipkavanja f_s.

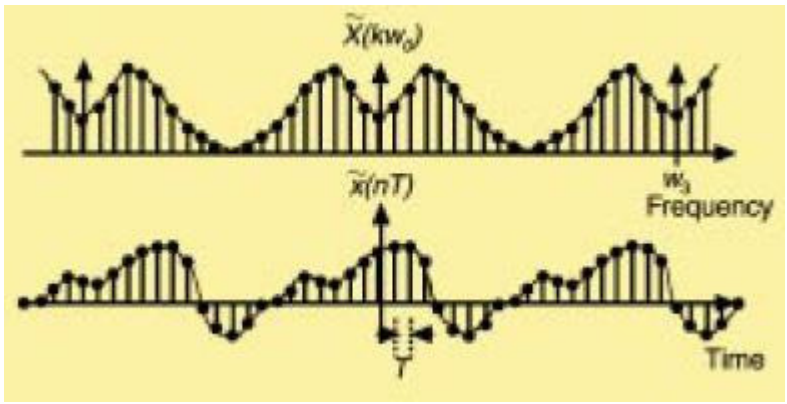


Slika 12. Spektar neperiodičnog diskretnog signala – preuzeo iz [4]

Slika 12 prikazuje kontinuirani spektar. Razlog tomu leži u množenju kontinuiranog signala x(t) s nizom Diracovih impulsa što rezultira konvolucijom u frekvencijskoj domeni između ta dva spektra (Diracovog niza i spektra signala). Proces otipkavanja detaljnije je razrađen u sljedećem poglavlju.

Možemo vidjeti da konvolucija zrcali spektar oko pola frekvencije otipkavanja, tj. spektar se ponavlja s periodom dužine frekvencije otipkavanja.

Sljedeći korak je diskretizacija spektra pošto računalo ne može raditi s kontinuiranim spektrom.



Slika 13. Diskretan spektar – preuzeto iz [4]

Slika 13 prikazuje uzorkovan spektar. Pogledamo li kako sada izgleda otipkani signal, primjećujemo da je on periodičan. Množenje spektra s Diracovim delta impulsima rezultira konvolucijom u vremenskoj domeni koja čini signal periodičnim. Važna osobina DFT algoritma je ta što očekuje da je ulazni signal periodičan. Vise u ovom problemu u poglavlju 4.5

Diskretiziranjem signala i spektra, postigli smo da su oba postala periodična i kao takvi predstavljaju beskonačan skup podataka.

Digitalnom računalu želimo ustupiti samo N uzoraka signala koji tvore jedan period. To se postiže jednostavnim množenjem s jediničnim prozorom koji pokriva N uzoraka (izvan perioda prozor ima vrijednost nula).

Sada kada imamo diskretan spektar $k\varpi_0$ i konačan broj uzoraka signala nT, možemo prilagoditi formule opisane u poglavlju 2.3.4 na sljedeći način.

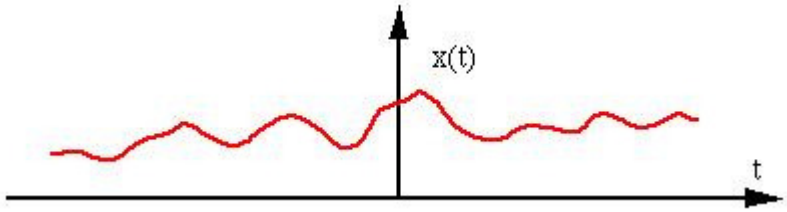
$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) e^{j(\frac{2\pi}{N})kn}$$

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j(\frac{2\pi}{N})kn}$$

Druga formula naziva se *Diskretna Fourierova transformacija* (DFT) i omogućava računalu izračun spektra signala od N otipkanih uzoraka. Prva formula predstavlja inverznu transformaciju spektra u signal.

2.3.6 Općenito o uzorkovanju

Uzorkovanje signala (otipkavanje) je postupak kojim iz vremenski kontinuiranog signala izvlačimo niz uzoraka (eng. *samples*). Promotrimo signal **x(t)** prikazan na slici 14.

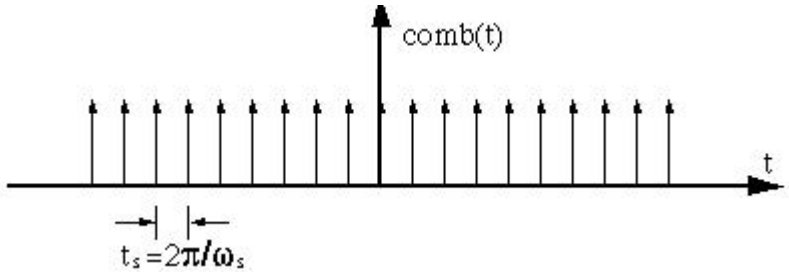


Slika 14. Kontinuirani signal

Da bismo izvukli uzorke, signal **x(t)** množimo sa funkcijom **comb(t)**.

$$comb(t) = \sum_{m=-\infty}^{\infty} \delta(t - mT_s)$$

Funkcija **comb(t)** se sastoji od niza diskretno postavljenih (po vremenu) D*iracovih delta* impulsa i prikazana je na slici 15.



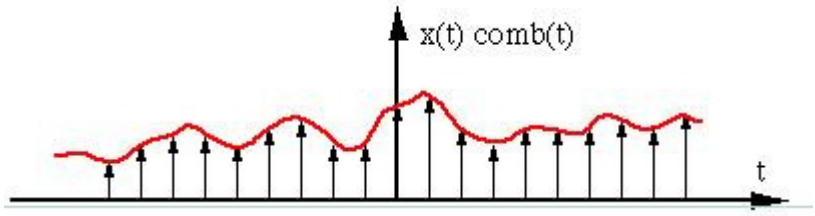
Slika 15. Niz *Diracovih* impulsa – comb(t)

Konstanta T_s predstavlja razmak između dva *Diracova* impulsa. Recipročna vrijednost od T_s, $1/T_s$ je vrlo bitan parametar, jer će nakon množenja jednoznačno odrediti **frekvenciju uzorkovanja** signala F_s.

Množimo **comb(t)** sa **x(t)**

$$x_u(t) = comb(t) \bullet x(t) = x(t) \sum_{m=-\infty}^{\infty} \delta(t - mT_s) = x(t) \mid_{t=mT_s}$$

Kao što vidimo, novonastali signal $x_u(t)$ odgovara početnom signalu isključivo u točkama uzorkovanja, dok je u ostalim točkama nula.



Slika 16. Novonastali signal

Crvena linija je početni signal. Vrijednosti u točkama mT_S odgovaraju *visini Diracovih impulsa* (zapravo odgovaraju površini impulsa, no radi lakšeg shvaćanja impulsi su skalirani po visini).

Signal $x_u(t)$ je i dalje kontinuiran (ima vrijednost nula između točaka uzorkovanja) i potrebno ga je diskretizirati na način da se samo *pokupe* dobiveni uzorci u točkama uzorkovanja.

$$x_u[n] = x_u(t)\,|_{t=mT_s} = x[0], x[1], x[2], ..., x[n]$$

Da bi dobiveni signal postao digitalan, potrebno ga je još i kvantizirati po amplitudi.

Postavlja se pitanje: Kada je moguće otipkani signal jednoznačno rekonstruirati? Odgovor na ovo pitanje daje *Shannonov* teorem o uzorkovanju, koji kaže da frekvencija uzorkovanja mora biti barem dvostruko veća od harmonika najveće frekvencije u signalu. Frekvencija najbržeg harmonika ponekad se naziva *Shannonova* frekvencija, za koju mora vrijediti spomenuto pravilo.

$$f_{shann} \leq \frac{1}{T_s}$$

Sve gore opisane postupke, u praksi radi analogno digitalni pretvornik (ADC) opisan u poglavlju 4.2.1

Brza Fourierova transformacija (FFT)

3.1 Uvod

Promotrimo na trenutak DFT izraz kako bismo utvrdili složenost algoritma. Za N-*point* DFT potrebno je napraviti N^2 kompleksnih množenja, tj. složenost iznosi približno $O(N^2)$. To je očito iz formule, jer za svaku izlaznu diskretnu frekvenciju, kojih ima N, moramo obaviti N kompleksnih množenja uzoraka ulaznog signala.

Točna složenost je nešto veća, jer se zanemaruje potrebna količina zbrajanja.

Kao primjer uzmimo da je N=512. Algoritam zahtijeva minimalno 262144 kompleksnih množenja !

Iako bi današnji (god. 2008) procesori s tim vjerojatno bez problema izašli na kraj, to bi značajno umanjilo važnost Fourierovog otkrića. Cilj je provoditi analizu spektra na vrlo štedljivim i samim time sporijim procesorima, kakvi se ugrađuju u ugradbena računala (npr. mali prijenosni analizatori spektra).

Iz tog su razloga matematičari i inženjeri razvili brojne metode ubrzanja algoritma, bez kojih ovaj završni rad ne bi bilo moguće napraviti na korištenom mikrokontroleru.

Svi algoritmi koji na neki način omogućuju digitalnom računalu brži izračun DFT-a zovu se **FFT** ili *Fast Fourier Transform* algoritmi.

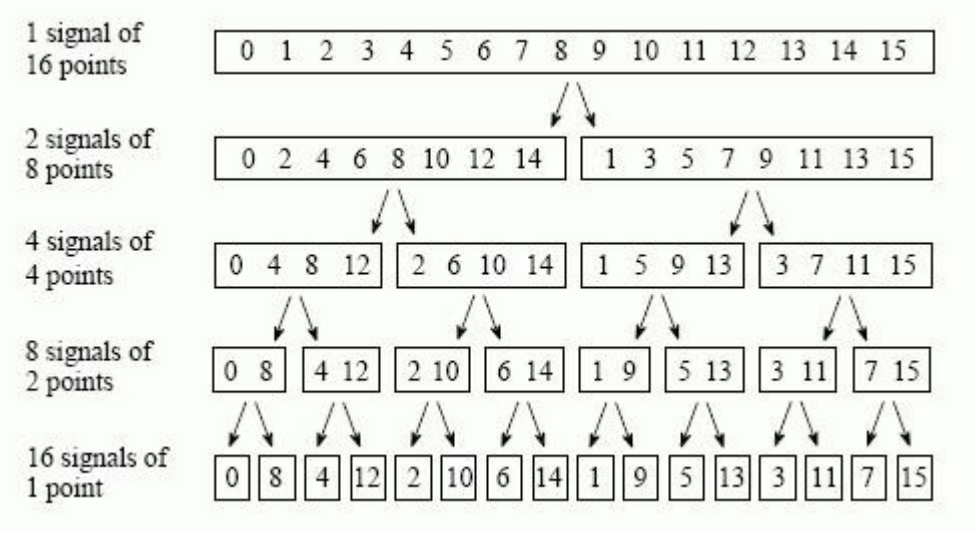
Jedan od njih je *Radix-2* algoritam i opisan je u nastavku, jer se upravo on koristi u ovom završnom radu.

3.2 Radix-2 algoritam

Radix-2 algoritam spada u najjednostavnije FFT algoritme.

Osnovni princip rada je da dijeli jednakost za diskretnu Fourierovu transformaciju (DFT) na dva dijela. Ta se novonastala dva dijela također dijele, sve dok ne ostane N signala svaki dužine jednog uzorka.

Ovaj je princip grafički prikazan na sljedećoj slici.



Slika 17. Princip rada Radix- 2 FFT – preuzeto iz [5]

Ovakav se algoritam naziva i algoritam s decimacijom u vremenu, jer preslaguje vremenske uzorke signala koji se obrađuje. Raspišimo sada DFT izraz po tom načelu.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}nk}$$

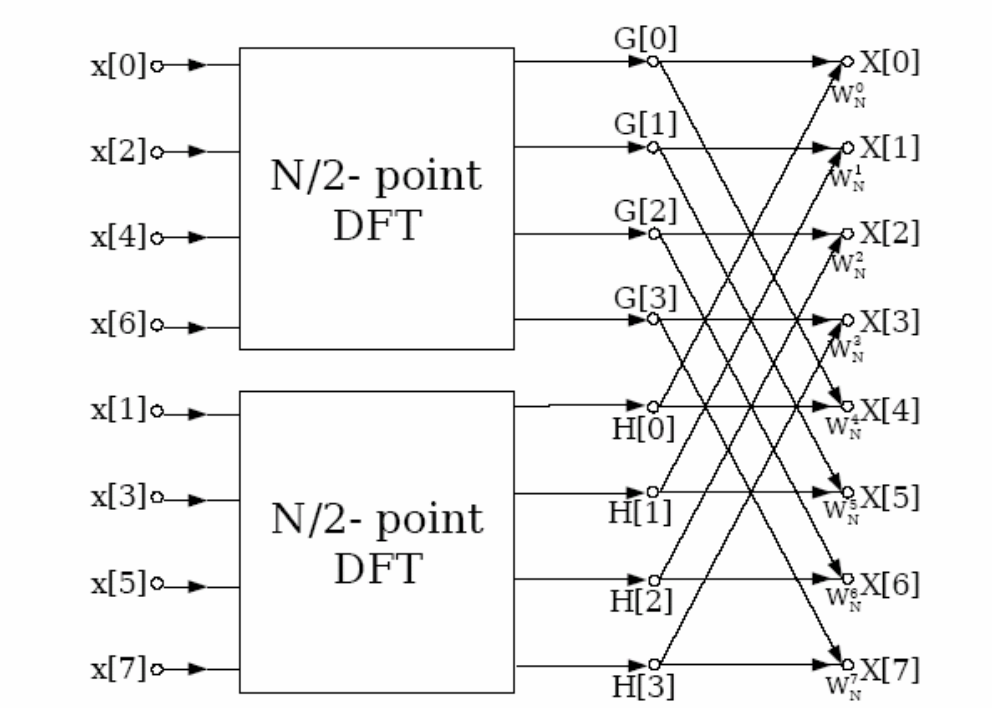
$$= \sum_{n=0}^{\frac{N}{2}-1} (x(2n) e^{-j\frac{2\pi}{N}(2n)k}) + \sum_{N=0}^{\frac{N}{2}-1} (x(2n+1) e^{-j\frac{2\pi}{N}(2n+1)k})$$

$$= \sum_{n=0}^{\frac{N}{2}-1} (x(2n) e^{-j\frac{2\pi}{\frac{N}{2}}nk}) + e^{-j\frac{2\pi}{N}k} \sum_{n=0}^{\frac{N}{2}-1} (x(2n+1) e^{-j\frac{2\pi}{\frac{N}{2}}nk})$$

$$= DFT_{\frac{N}{2}}[x(x), x(2), ..., x(N-2)] + W_N^k DFT[x(1), x(3), ..., x(N-1)]$$

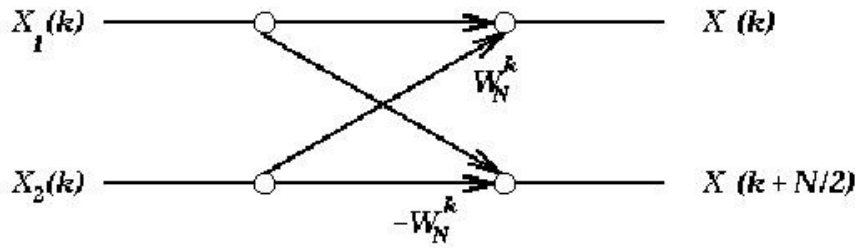
Gornja jednakost pokazuje da se izlazna spektralna komponenta može izračunati kao suma dva manja DFT-a dužine N/2. Pri tome su uzorci podijeljeni u grupe s parnim i neparnim indeksima.

Neparni dio DFT-a množi se s faktorom W_N^k koji se u literaturi često naziva *twiddle faktor.*



Slika 18. Grafička interpretacije algoritma – preuzeto iz [5]

S G(k) su označeni frekvencijski izlazi iz prvog parnog DFT-a, dok su sa H(k) oni neparnog. Zbog periodičnosti frekvencijskih uzoraka s N/2, kombinirajući izlaze dva manja DFT-a, moguće je izračunati dva frekvencijska uzorka (izlaza) N-*point* DFT-a uz korištenje različitih *twiddle* faktora.



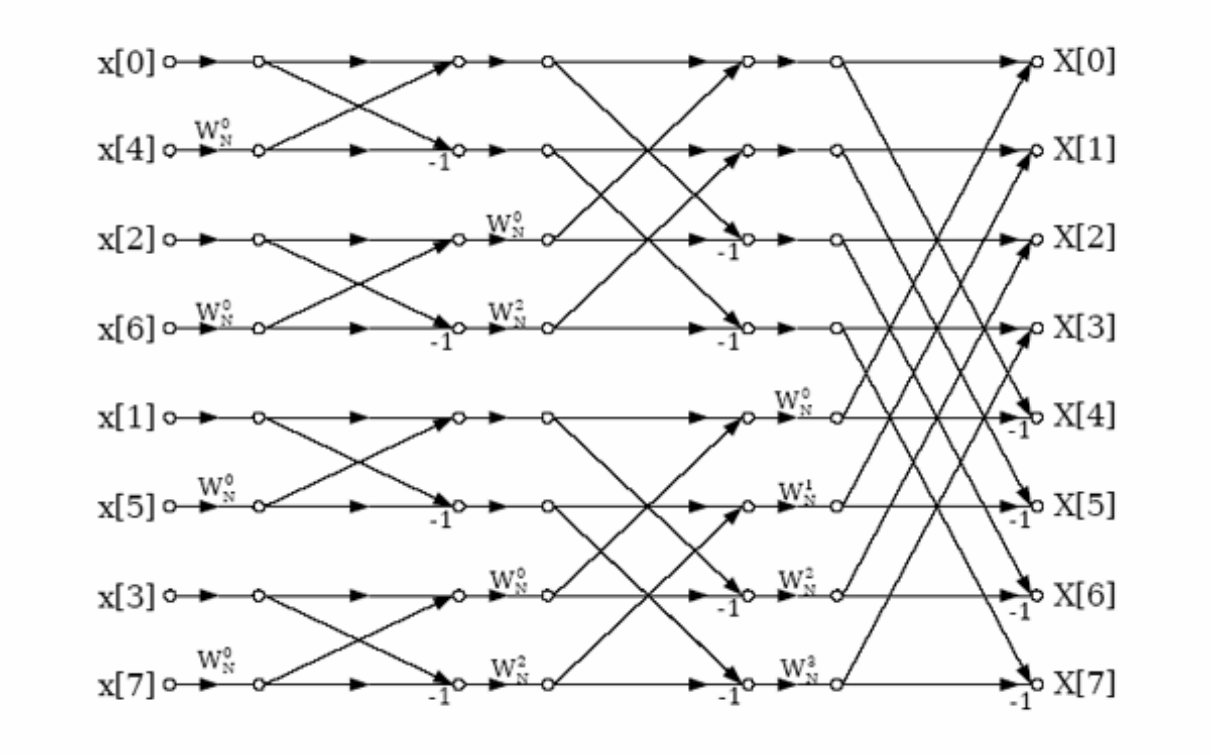
Slika 14. *Butterfly* kompleksno množenje

Kombiniranje uzoraka manjih DFT-a u veći, prikazano je na slici 14. Dobilo je ime *butterfly* zbog oblika protoka podataka koji podsjeća na leptira. *Butterfly* se sastoji od sljedeće dvije jednadžbe.

$$X(k) = x_1(k) + W_N^k x_2(k)$$

$$X(k + \frac{N}{2}) = x_1(k) - W_N^k x_2(k)$$

Kada se nad skupom od N ulaznih podataka primjeni algoritam do kraja (dok se podaci ne sortiraju u grupice od dva uzorka) i tijek izračuna spektra prikaže gore opisanim butterfly jednadžbama, dobije se dijagram toka prikazan na slici 15.



Slika 19. Ukupan dijagram toka Radix- 2 algoritma nad 8 uzoraka – preuzeto iz [5]

Napomena: slika demonstrira izračun spektra za 8 uzoraka, a ne 16, kao na slici 13, isključivo zbog preglednosti. Princip je isti.

Iz njega se vidi da se algoritam izvršava po nivoima (eng. *stage*). Ukupan broj nivoa iznosi $\log_2 N$. U svakom nivou obavi se točno N/2 butterfly izračuna. Jedan butterfly izračun uključuje jedno kompleksno množenje i dva zbrajanja (osvrnite se na jednadžbe na prethodnoj stranici, *twiddle* faktor se samo jednom množi s x2(k)). Prema tome ukupna složenost FFT Radix-2 DIT algoritma iznosi

$$\frac{N}{2}\log_2 N$$

Direktan izračun spektra pomoću DFT algoritma za N=512 uzoraka zahtijevao bi 262144 kompleksnih množenja (poglavlje 3.1). Koristeći FFT algoritam, taj se broj

svodi na samo 2560 množenja. Što je veći broj uzoraka, to FFT više pokazuje svoju brzinsku prednost nad direktnim izračunom DFT algoritmom.

Pogled na sliku 15 otkriva da se ulazni uzorci moraju na neki način preurediti, kako bi se dobili izlazi u pravilnom redoslijedu. Vrlo je lako uočiti da se svaki uzorak nalazi točno na obrnutom (eng. *reversed*) indeksu. Zamjena indeksa uzoraka prikazana je u tablici 1.

Tablica 1. Redoslijed uzoraka na ulazu u FFT

Indeks dekad.	Indeks (baza 2)	Obrnuti index	Obrnuti (baza 10)
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

Uzorak signala koji se u trenutku uzorkovanja nalazio na indeksu 1 ($001_{(2)}$) prelazi u indeks ($100_{(2)}$), tj. 4 itd.

Samo preslagivanje uzoraka na ovaj način zahtijeva određeno procesorsko vrijeme. Prije njegove implementacije potrebno se zapitati: Je li ono stvarno potrebno? Npr. ukoliko aplikacija služi za detektiranje maksimalne frekvencije u signalu, preslagivanje na ulazu u FFT nije potrebno. Softver može nakon izračuna spektra pretražiti rezultate i okrenuti samo onaj indeks na kojemu se nalazi maksimalna vrijednost amplitude.

S obzirom na to da je cilj ovog završnog rada izrada analizatora spektra u kojem su sve frekvencijske komponente bitne, okretanje indeksa svih ulaznih uzoraka izvršit će se prije *propuštanja* uzoraka kroz FFT algoritam.

Analizator spektra

4.1 Koncept i specifikacija uređaja

Kao što je već spomenuto u uvodu, cilj ovog završnog rada je izrada analizatora spektra audio signala.

Prije bilo kakvog početka rada na samoj realizaciji analizatora, bitno je definirati, tj. zamisliti koncept samog uređaja.

Koncept prije početka izrade bio je stvoriti vrlo malen, atraktivan i jeftin analizator, koji će biti sposoban obrađivati ulazni signal u realnom vremenu.

Iz tog vrlo šturog i jednostavnog koncepta izvučeni su zahtjevi, tj. specifikacije koje se nameću na korišteno ugradbeno računalo, pri čemu se zahtjevi uglavnom odnose na izbor mikrokontrolera.

Prema tome, korišteno ugradbeno računalo mora zadovoljiti sljedeće:

- mikrokontroler mora biti dovoljno brz da izračuna 1024-*point* FFT u realnom vremenu
- mora imati AD pretvornik s rezolucijom od barem 10-bita te **SH** sklopom
- mora imati *Timer-Counter* kako bi se u determiniranim vremenskih razmacima uzorkovao ulazni signal
- mora imati SPI sinkronu sabirnicu ugrađenu u raspoloživu periferiju radi komunikacije s *Dot Matrix Display-om* (tako da nije potrebna softverska emulacija)
- kako bi se maksimalno ubrzao i pojednostavio softver poželjna je pristupnost *Direct Memory Access* (**DMA**) sklopa
- arhitektura mora biti jednostavna, a dostupni programski alati besplatni i kvalitetni
- sigurna mogućnost nabave svih potrebnih komponenti od strane lokalnih distributera (Farnell u prvom redu)
- na raspolaganju razvojni sustav kako bi što više pojednostavnio i ubrzao razvoj

U trenutku izrade ovog završnog rada Zavod je imao na raspolaganju nekoliko razvojnih sustava s *Atmelovim* **AT91SAM7X256** mikrokontrolerima, koji u potpunosti zadovoljavaju gore navedene specifikacije.

Dotični razvojni sustav pruža korisniku USB programiranje, RS-232 vezu s računalom, priključenje ULINK uređaja na JTAG, LAN vezu te omogućuje pristup svim digitalnim kao i analognim priključcima mikrokontrolera putem 100mil *header* konektora.

Više o razvojnom sustavu može se pročitati u [7].

4.2 Mikrokontroler

AT91SAM7X256 spada u AT91**SmartARM** familiju Atmelovih mikrokontrolera.

Kao što ime familije kaže, baziraju se na **ARM7TDMI** jezgri. ARM7 je bez konkurencije najkorišteniji i najpopularniji 32-bitni procesor korišten u ugradbenim računalima.

Zbog toga nije ni čudno što se Atmel odlučio upravo za njega u jednoj od svojih po mogućnostima najjačoj familiji mikrokontrolera. Drugim riječima, AT91SAM7X spada u sam *high end* segment tržista mikrokontrolerima.

Čitatelju se može činiti da se odabir dotičnog mikrokontrolera kosi s samim konceptom *jeftinog* analizatora opisanog u prošlom poglavlju. Treba znati da najveći udio na tržištu čine vrlo jeftini 8-bitni mikrokontroleri, koji svojim mogućnostima ne odgovaraju zahtjevima analizatora spektra.

Zbog tog se razloga AT91SAM7X256 s cijenom od 150kn (sredina 2008. godine) dobro uklapa u koncept projekta s odličnim omjerom cijena/mogućnosti (performanse).

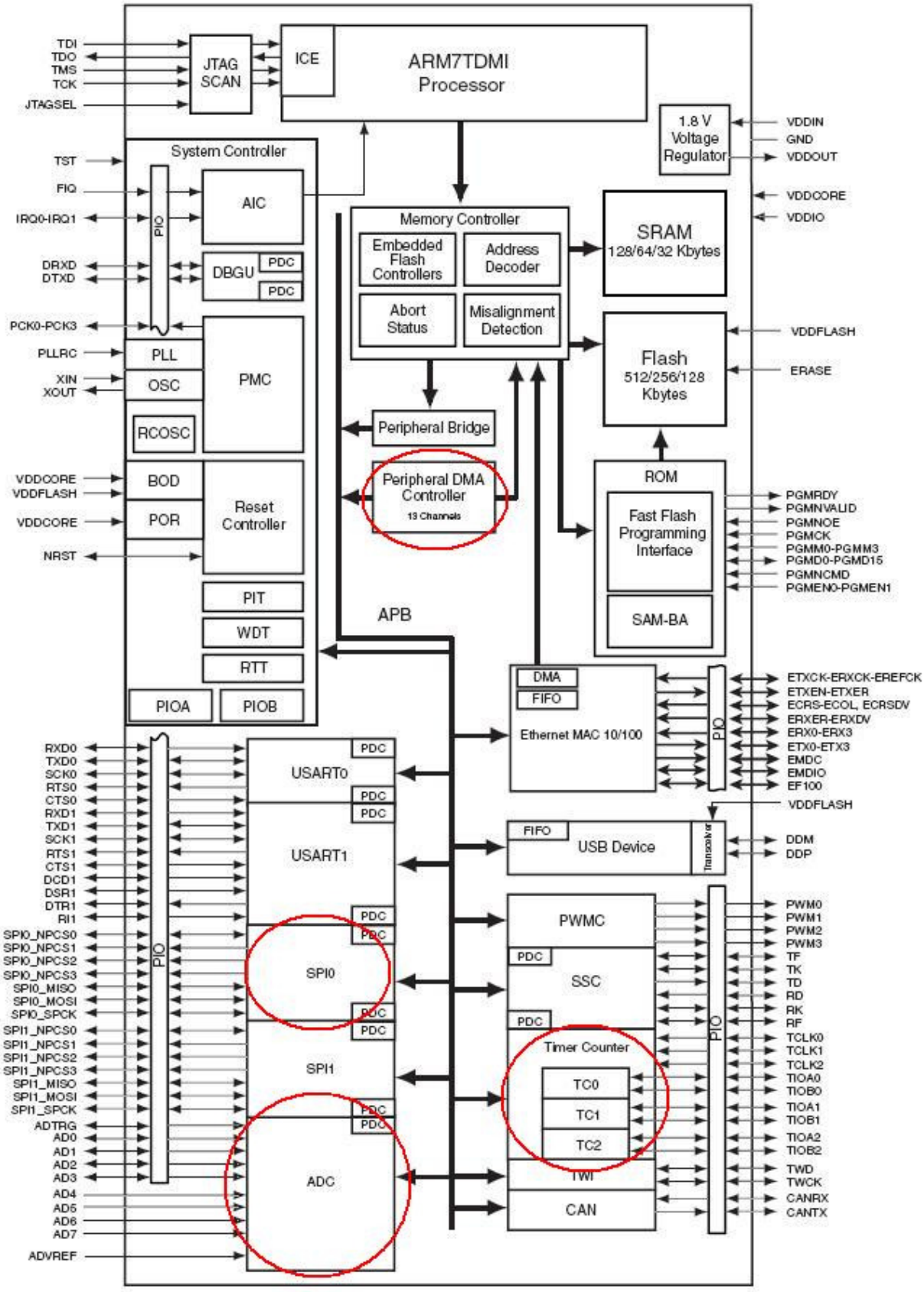
Glavne karakteristike korištenog mikrokontrolera popisane su u nastavku:

- ARM7TDMI 32-bitni RISC procesor
- 256kB interne brze FLASH memorije (ROM)
- 64kB SDRAM memorije
- Memorijski kontroler (*Memory Controller*)
- RESET kontroler (*Reset Controller*)
- Kontroler za obradu prekida (*Advanced Interrupt Controller*)
- Generator takta (PLL)
- Upravljanje potrošnom sklopa (*Power Management Controller*)
- USB 2.0, I2C (TWI), SPI, UART, LAN, CAN, SSC
- 3 indentična 16-bitna brojila
- Samostalni PWM kanali
- 10-bitni ADC s **S**ample and **H**old sklopom
- Sklop za direktan pristup memoriji, PDC (*Peripheral Direct Memory Access Controller*)

Kompletan popis svih mogućnosti, kao i njihov detaljan opis, može se pronaći u [6]. Navedene glavne karakteristike poklapaju se s zahtjevima projekta.

Za programiranje ove familije kontrolera, dostupan je programski alat ***Keil uVision.*** U besplatnoj inačici pruža potpunu funkcionalnost uz jedino ograničenje na maksimalnu količinu programske memorije od 16kB, što za ovaj projekt ne predstavlja stvarno ograničenje.

Figure 2-1. AT91SAM7X512/256/128 Block Diagram



Slika 20. AT91SAM7X256 Blok dijagram – preuzeo iz [6]

Slika 16. prikazuje blok dijagram mikrokontrolera. S crvenom bojom označeni su dijelovi periferije korišteni u izradi analizatora spektra.

- ADC -> Analogno digitalni pretvornik
- Timer Counter -> Brojilo
- Peripheral DMA Controller -> Sklop za direktan pristup memoriji
- SPI0 -> Serijska sinkrona sabirnica

Analogno digitalni pretvornik, zajedno s brojilom i DMA sklopom služe za uzorkovanje ulaznog signala. Kratak opis njihovog rada i način korištenja nalazi se u sljedeća tri poglavlja.

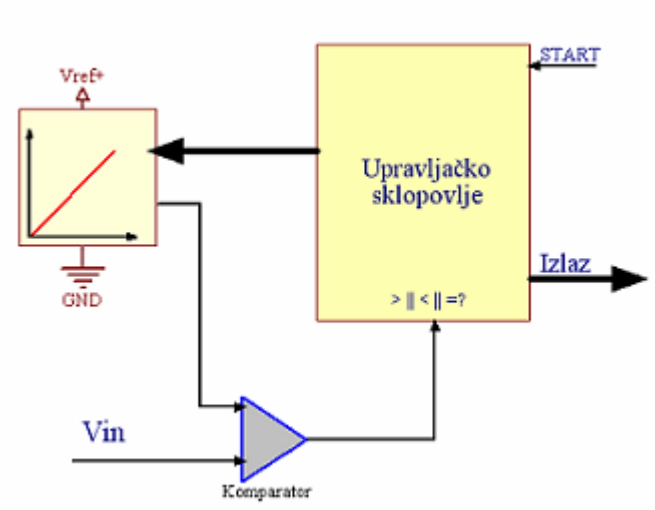
SPI sabirnica služi isključivo za komunikaciju s displayom i opisana je u poglavlju 5.4.

4.2.1 Analogno digitalni pretvornik

AD pretvornik je analogno-digitalni sklop, koji ulazni kontinuirani signal (napon ili struju) pretvara u niz digitalnih brojeva. Postoji više načina na koji je to u praksi moguće ostvariti, no tu će biti opisan samo jedan postupak, na kojem je zasnovan AD pretvornik korišten u ovom projektu.

U mikrokontroleru AT91SAM7X256 nalazi se 10-bitni AD pretvornik zasnovan na **uzastopnoj aproksimaciji (*eng. successive aproximation*)**.

Za vrijeme trajanja pretvorbe, ulazni signal se konstantno uspoređuje s promjenjivim internim referentnim signalom. Referenti signal je diskretan i najčešće izveden višestrukim otpornim dijelilom između mase i potencijala V_{ref}, tj. maksimalne vrijednosti koju AD može pretvoriti.



Slika 21. Blok shema AD pretvornika s sukcesivnom aproksimacijom

Pretpostavimo da ulazni signal u trenutnu uzorkovanja iznosi 2V. Referenti napon je podešen na 3V. AD će obavit konverziju maksimalnom brzinom, a to znači da mora koristiti postupak binarnog pretraživanja.

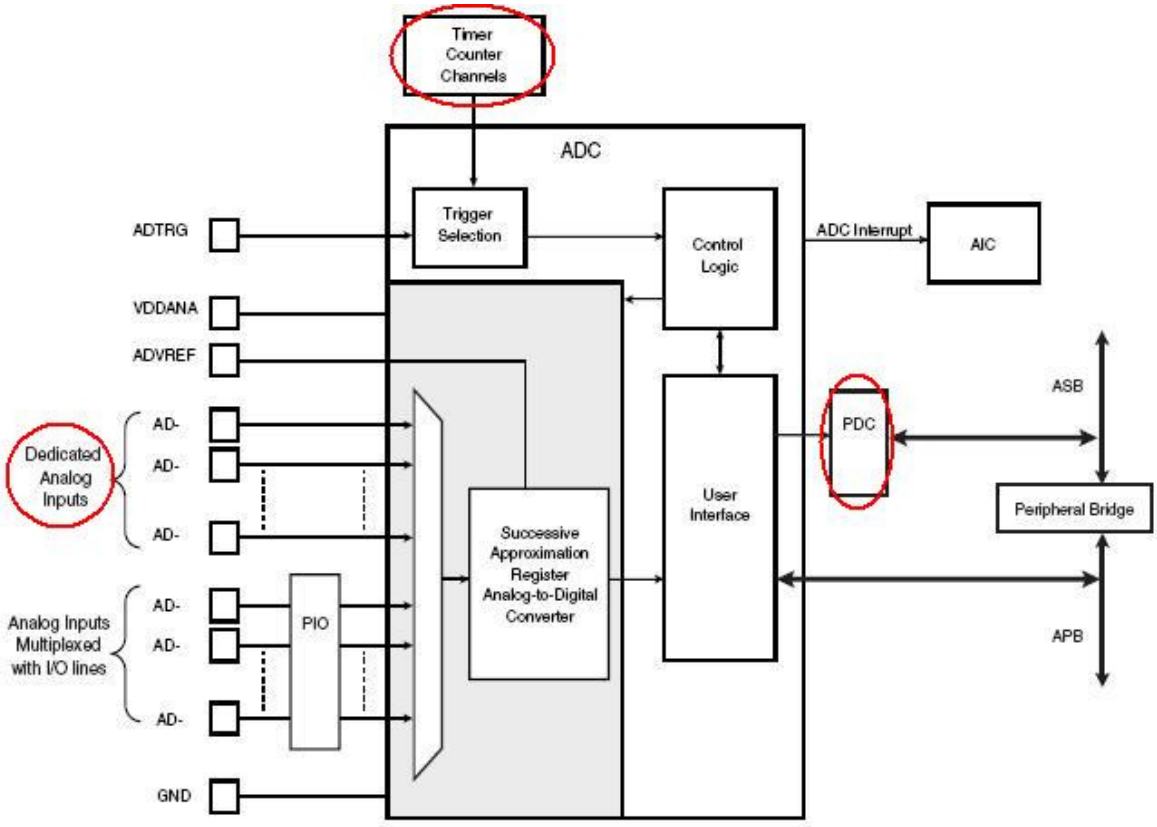
U prvoj aproksimaciji, uspoređuje ulazni signal s $V_{ref}/2$. Pošto je 2>1.5, MSB se postavlja (**M**ost**S**ignificant**B**it) u $1_{(2)}$.

U sljedećoj aproksimaciji, ostatak područja gdje se rezultat može nalaziti, opet se dijeli na dva dijela i vrši usporedba 2>2.25. Pošto izraz nije točan, trenutni izlaz je

sada $10_{(2)}$. Ovaj postupak se ponavlja N puta, gdje N predstavlja rezoluciju AD pretvornika.

Razmotrimo malo koliko je vremena potrebno AD-u da na ovaj način obavi jednu pretvorbu. Zna se da binarno pretraživanje rezultira uspjehom u najviše $\log_2 M$ puta gdje M predstavlja broj diskretnih razina na koje AD pretvornik može podijelit područje ulaznog signala (0-V_{ref}) i iznosi upravo 2^{10}. 10 je rezolucija korištenog AD pretvornika. Drugim riječima, broj strojnih ciklusa AD pretvornika potreban za jednu konverziju iznosi $\log_2 2^{10} = 10$.

Za potrebe analizatora spektra audio signala, ulazni signal će se otipkavati frekvencijom od 44.1kHz. Minimalan takt na ulazu u AD pretvornik iznosi $F_s \bullet 10$, tj. 441kHz, što ne predstavlja nikakav problem s obzirom na korišteni pretvornik.



Slika 22. Blok shema AD pretvornika u AT91SAM7X256 – preuzeto iz [6]

Slika 22 prikazuje blok dijagram AD pretvornika u AT91SAM7X256.

Moguće je vršiti konverziju u rezoluciji 8 ili 10 bita na ukupno 8 ulaznih analognih kanala, koji se pomoću 8 na 1 analognog multipleksora dovode na sam sukcesivni pretvornik. Od dostupnih 8 kanala, pola (4) ih je multipleksirano s PIO sklopom, dok su ostali namjenski (eng. *dedicated*).

S obzirom da je za analizator potreban samo jedan kanal, odabran je prvi namjenski kanal imena **CH4**, tj. Channel 4.

Da bi došlo do konverzije na omogućenom kanalu, potrebno je okinuti (eng. *trigger*) sam pretvornik da započne konverziju. Okidanje može biti hardversko ili softversko.

Softversko okidanje omogućuje procesoru da započne konverziju u bilo kojem trenutku. Hardversko okidanje je moguće eksterno preko priključka ADTRG, ili pak interno pomoću bilo kojeg od 3 postojeća brojila.

Obzirom da se ulazni signal mora uzimati u točno determiniranim vremenskih razmacima, ADC je podešen za vršenje konverzija isključivo na hardversko interno okidanje uzrokovano impulsom iz *Timer Counter* 0 sklopa.

Podešavanje dotičnog brojila i kratak opis rada, nalaze se u sljedećem poglavlju.

Vrlo je bitno ne prekidati procesor nakon što se prikupi N uzoraka signala, koji se odmah zatim obrađuju. AT91SAM familija mikrokontrolera omogućuje direktnu vezu između SDRAM memorije i većine periferije putem DMA sklopa.

Svaki put kad brojilo pošalje okidni signal analogno digitalnom pretvorniku, on će nakon što pretvorba završi, kontaktirati DMA sklop koji će bez ikakvog prisustva procesora prebaciti rezultat u memoriju. Nakon što se prikupi svih N uzoraka signala, DMA sklop obavještava procesor da su novi uzorci spremi za obradu.

Opis podešavanja DMA sklopa nalazi se u poglavlju 4.2.3.

Rezimirajući sve navedeno, analogno digitalni pretvornik podešen je na sljedeći način:

- eksterno okidanje iz *Timer Counter Channel 0* (prvog brojila)
- rezolucija pretvorbe 10 bita

- normalan način rada (između pretvorbi ADC neće ulaziti u *sleep* jer bi mu u trenutku dolaska okidanja bilo potrebno određeno minimalno vrijeme buđenja)
- radni takt ADC logike podešen na 532.48kHz
- *startup* vrijeme 480us
- ***S**ample and **H**old* vrijeme 4us
- omogućen kanal 4 (namjenski analogni ulaz koji nije vezan za PIO kontroler)

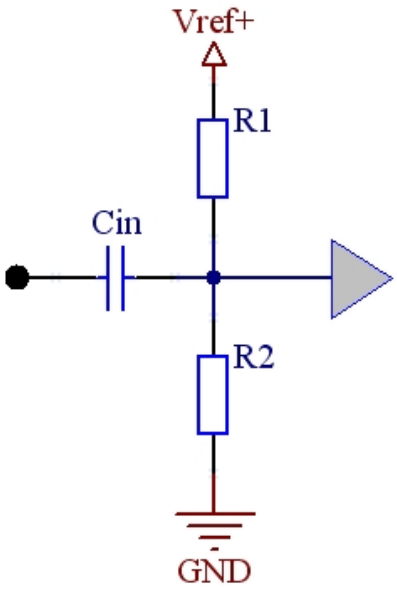
Programski kôd koji izvršava navedeno, prikazan je u nastavku. Za točno značenje korištenih ***S**pecial **F**unction* **R**egistara (**SFR**) potrebno je proučiti [6], poglavlje 35.

```
#define TRGSEL              0
#define ADCLOCK          0x02
#define STARTUP          0x1F
#define SHTIM            0x04

void ADC_Init()
{
      ADC_MR=(1<<TRGSEL)|(ADCLOCK << 8)|(STARTUP << 16)|(SHTIM << 24);
      ADC_CHER=(1<<CH4);
}
```

4.2.2 Ulaz signala

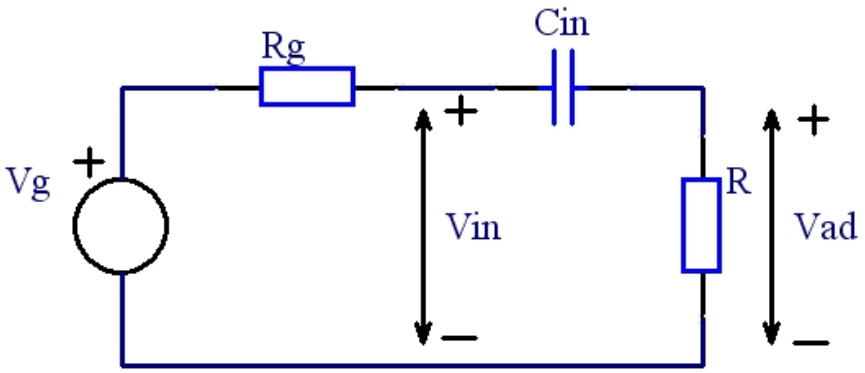
Kako bi se signal mogao uzorkovati, potrebno ga je prvo kvalitetno dovesti na ulaz AD pretvornika. Korišteni AD pretvornik je unipolarnog tipa, što znači da je za priključenje bipolarnog audio signala potrebno prvo namjestiti radnu točku. To se postiže jednostavnim sklopom prikazanim na slici 23.



Slika 23. Ulaz signala

U statičkim uvjetima, kondenzator Cin predstavlja prekid. Otpornici R1 i R2 čine naponsko djelilo. Cilj je omogućiti maksimalni hod ulaznog signala, pa se izabiru jednaki otpornici, tj. R1=R2 što postavlja radnu točku na pola napona V_{ref}, tj. 1.65V (posto je V_{ref} 3.3V)

Osim podešavanja radne točke, sklop sa slike predstavlja i visoko propusni filtar. Naime, nije moguće izbjeći ulazni kondenzator, jer bi otpor generatora signala poremetio radnu točku.



Slika 24. Ulaz signala – dinamika

Na slici 24 prikazan je isti sklop u dinamičkim uvjetima. Otpornici R1 i R2 su spojeni paralelno i zamijenjeni otpornikom R=R1||R2, jer je V_{ref} u dinamici na potencijalu mase. Na ulaz sklopa je priključen signal iz MP3 *playera*. Taj spoj unosi u sklop naponski izvor V_g zajedno s izlaznim otporom pojačala R_g.

Treba napomenuti, da je paralelno otporniku R spojena ulazna impedancija AD pretvornika (otpor reda veličine 10^7 i kondenzator reda veličine 10^{-12}), no ona se može zanemariti za frekvencije nama od interesa.

V_{ad} je napon kojeg *vidi* AD pretvornik. Ulazni napon V_{in} se dijeli između kondenzatora Cin i otpornika R. Drugim riječima, kondenzator će prigušiti signale niske frekvencije. Ovo nam daje mogućnost da podesimo vrijednosti kondenzatora i otpornika R, tako da donja granična frekvencija ulaza bude oko 15Hz (audio signal koji nas zanima se nalazi između 20-20000Hz).

Prijenosna funkcija ulaza je

$$\frac{V_{ad}}{V_{in}} = \frac{R}{R + \frac{1}{j\varpi C_{in}}} = \frac{j\varpi RC_{in}}{1 + j\varpi RC} = \frac{j\frac{\varpi}{\varpi_1}}{1 + j\frac{\varpi}{\varpi_1}},\ \varpi_1 = \frac{1}{RC_{in}}$$

Vidimo da prijenosna funkcija unosi jednu nulu na nuli i pol na ϖ_1. ϖ_1 upravo predstavlja donju graničnu kružnu frekvenciju.

Iz ovoga slijedi da je donja granična frekvencija jednaka

$$f_d = \frac{1}{2\pi RC_{in}}$$

Kondenzator se može odabrati proizvoljno (treba uzeti u obzir dostupnost i red veličine moguć za keramičke kondenzatore) i neka iznosi 1uF. Iz toga slijedi da otpor mora biti 10.6kΩ, tj. R1=R2=2R=21.2kΩ.

Treba razmotriti koji utjecaj ima izlazni otpor generatora na gušenje signala. Prijenosna funkcija koja opisuje njegov utjecaj je

$$\frac{V_{ad}}{V_g} = \frac{Z_{ul}}{Z_{ul} + R_g}$$

Iz nje je jasno da će ulazni signal biti kvalitetniji (manje prigušen), što je ulazna impedancija veća i izlazni otpor pojačala (generatora signala) manji. Na izlazni otpor ne možemo utjecati i njegova vrijednost je sigurno blizu otpora samih slušalica (radi optimalnog prijenosa snage) tj. otprilike 5-15Ω (slušalice imaju

tipično otpor 8Ω). Na ulaznu impedanciju možemo utjecat odabirom otpora i kapaciteta.

Ona iznosi

$$Z_{ul} = \frac{1 + j\omega RC}{j\omega C}, \quad |Z_{ul}| = \frac{\sqrt{1 + (\omega RC)^2}}{\omega C}$$

Jasno je da će se modul te impedancije kretati od beskonačno za frekvenciju nula do $10.6k\Omega$, koliko iznosi otpor otpornika (obzirom da su spojeni u seriju). Na frekvenciji od 15Hz (što je najgori slučaj signala koji nas zanima) gubitak signala na R_g iznosi 667μV/V što je zanemarivo. Ono što može predstavljati problem je promjena tog gubitka s frekvencijom. Već za malo veće frekvencije od 15Hz, ulazna impedancija postaje (skoro) konstantna i iznosi $10.6k\Omega$ tako da je faktor promjene impedancije na različito gušenje s frekvencijom zanemariv.

4.2.2 Brojilo

Kako bi se postigla stalna i točna frekvencija uzimanja uzoraka, koristi se jedno od tri, 16-bitnih brojila, kojima raspolaže sam mikrokontroler.

Kao mali podsjetnik iz prošlog poglavlja, AD pretvornik je podešen tako da započinje konverziju isključivo na okidni signal (eng. *trigger*) koji na svome izlazu generira brojilo.

Brojilo treba programirati na način da generira periodičan signal frekvencije 44100Hz, jer upravo toliko iznosi frekvencija uzorkovanja.

AD pretvornik se na taj signal okida isključivo na rastući brid, pa radi toga radni omjer izlaznog signala (eng. *duty cycle*) nije bitan.

Kratkim pogledom na *datasheet* otkriva se da se brojilo može programirati tako da radi na dva načina.

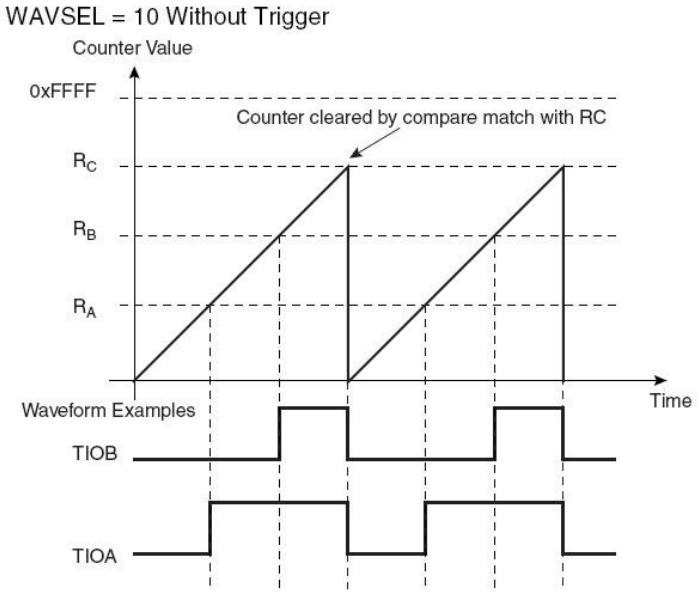
Prvi je «C*apture mode*» način, u kojem se brojilo obično koristiti za brojanje vanjskih (eksternih) impulsa, izračun faze i frekvencije signala, protek vremena između dva događaja i slično.

Drugi se način naziva «*Waveform mode*» i namijenjen je kao što mu samo ime govori generiranju valnih oblika (pravokutnog) signala. Lako je zaključiti da brojilo treba podesiti u waveform način rada.

U waveform načinu rada programer određuje oblik izlaznog periodičnog signala. Pri tome se misli na frekvenciju i radni omjer.

Radni omjer se podešava definirajući trajanje visoke i/ili niske razine signala. Kao što je već spomenuto, radni omjer nije bitan za rad AD pretvornika.

Sam waveform način rada omogućuje odabir nekoliko različitih tipova rada. Potrebno je podesiti brojilo tako da se *poništava* nakon proizvoljnog (period uzorkovanja) vremena brojanja.



Slika 25. Generiranje signala na TIOA – preuzeto iz 6

Slika 19. prikazuje način rada u kojem je WAVSEL podešen na 0x02.

Brojilo ima na raspolaganju tri registra koja se neprekidno uspoređuju sa samim sadržajem brojila. Kada se dogodi jednakost bilo koja od navedena tri registra, mogu se poduzeti određene akcije nad izlaznim signalom (npr. digni visoko, spusti nisko, invertiraj stanje, ne radi ništa).

Kao primjer na slici se vidi da signal **TIOA** (koji je doveden na AD pretvornik) prelazi u nisko stanje na jednakost RC registra i brojila, a visoko kada se broj u brojilo izjednači s brojem u registru RA.

Sa slike 19 je vidljivo da registar C određuje frekvenciju izlaznog signala, jer se upravo na njega brojilo poništava i kreće brojiti od početka.

Izgled izlaznog signala, ovisno o stanju usporedbi, podešava se proizvoljno i ne mora izgledati kao prikazan na slici.

Za potrebe okidanja AD pretvornika, TIOA signal dizati će se visoko, svaki put kad se brojilo poništi (RC usporedba). Vrijednost registra RC je odabrana tako da na frekvenciji brojanja izjednači vrijednost s brojačem nakon točno 1/44100 sekundi.

Nije bitno kada će se signal spustiti nisko, bitno je samo da se spusti. Registar RA će spuštati signal nakon 1/(2*44100) vremena. TIOA signal koji okida AD pretvornik nakon ovih podešavanja, poprima otprilike pravilan pravokutan oblik s radnim omjerom 50%.

AT91SAM familija mikrokontrolera omogućuje inženjerima strogo i vrlo predvidivo određivanje potrošnje samog sklopa. Za to je zaslužan dio periferije pod imenom ***P**ower **M**anagement **C**ontroller* (**PMC**). On, između ostalog, određuje koji će dijelovi samog mikrokontrolera dobivati radni takt, a koji ne.

Tu se prvotno misli na periferiju, no moguće je isključiti radni takt i samom procesoru kad cijeli mikrokontroler prolazi u poseban vrlo štedljiv način rada.

Da bi se uopće omogućio rad brojila, potrebno je prvo podesit **PMC** tako da opskrbljuje brojio radnim taktom.

Čitatelj možda primjećuje da AD pretvorniku nije bilo potrebno omogućavati radni takt (nakon *reseta* sva je periferija ugašena). AD pretvornik je jedina iznimka, tj. jedini podsklop mikrokontrolera na kojeg **PMC** nema utjecaja.

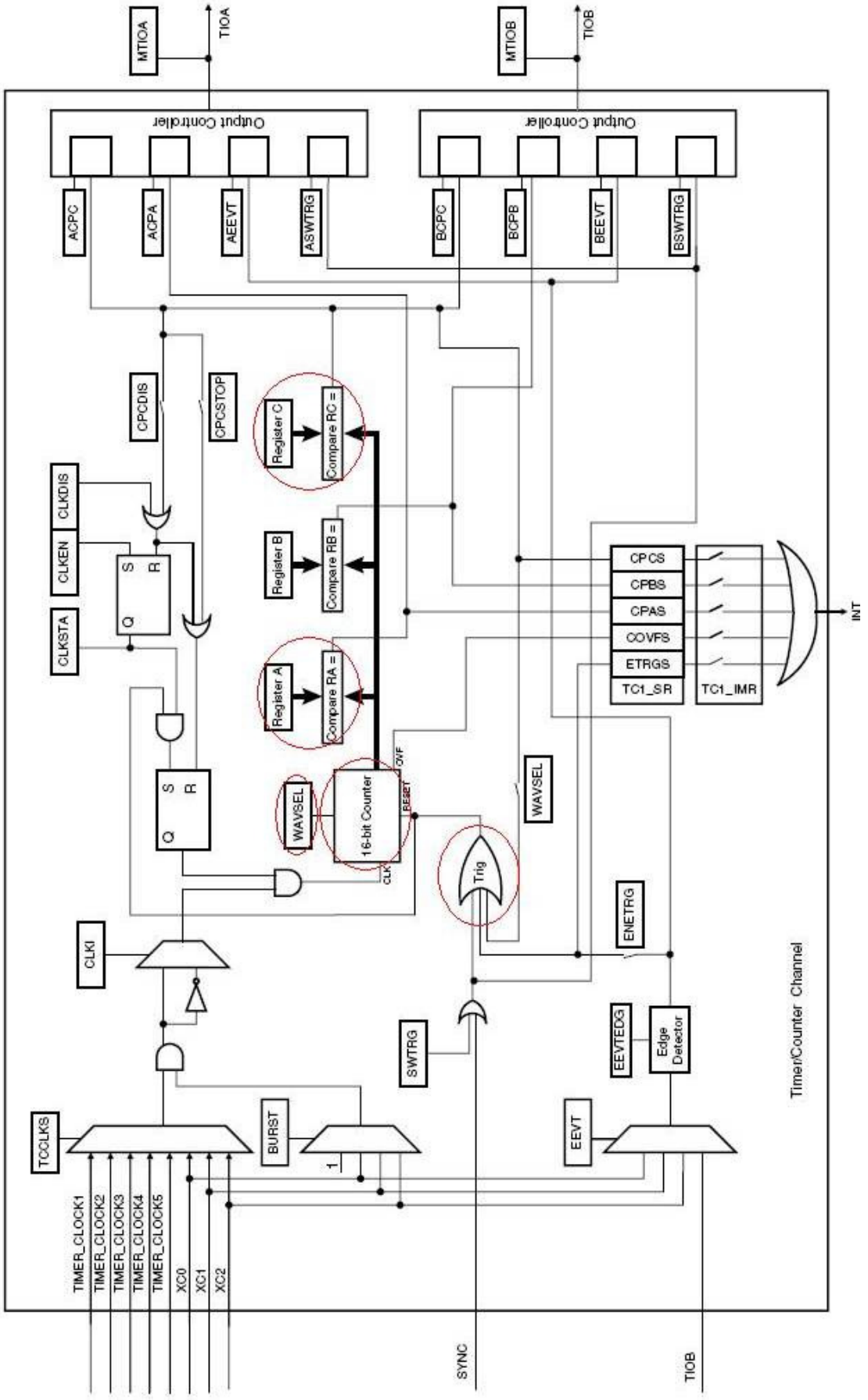
Slika 20. prikazuje nešto detaljniju blok shemu jednog od tri brojila (sva tri su identična). Crvenoj bojom su zaokruženi dijelovi spominjani u ovom odjeljku. Da bi brojilo ispravno radilo, potrebno je podesiti i nekolicinu ostalih *sitnih* stvari koje zainteresirani čitatelj može sam proučiti iz ustupljenog programskog odsječka.

U nastavku je prikazan programski odsječak koji izvršava svu potrebnu inicijalizaciju brojila u način rada opisan u ovom poglavlju.

Za točno značenje korištenih **SFR** registara, potrebno je proučiti [6], poglavlje 32.

```
#define TIMER_CLOCK2      0x01
#define WAVESEL           0x02
#define ACPA              0x02
#define ACPC              0x01
#define CLKEN             0x01
#define SWTRG             0x01

void TC0_Init(void)
{
       PMC_PCER|=(1<<12);
       TC0_CMR=(1<<15);
       TC0_CMR|=(TIMER_CLOCK2 << 0)|(WAVESEL << 13)|(ACPA << 16)|(ACPC << 18);
       TC0_RA=65;
       TC0_RC=136;
       TC0_CCR=(CLKEN << 0)|(SWTRG << 2);
}
```



Slika 26. Blok shema brojila za *waveform* rad – preuzeto iz [6]

4.2.3 Direct Memory Access (DMA) sklop

Unatoč velikom ubrzanju koje pruža **FFT** algoritam pri izračunu spektra, spektralna analiza je i dalje vrlo zahtjevna operacija. Ukoliko se želi postići izračun spektra u realnom vremenu, svaki komadić procesorskog vremena je dragocjen, a nepotrebne instrukcije nepoželjne.

To posebno vrijedi za izračun spektra na procesoru (**ARM7TDMI**) koji se koristi u ovom projektu. Specijalizirani **D**igital **S**ignal **P**rocessor (**DSP**) procesori imaju prilagođenu arhitekturu ne bi li s velikom brzinom računali FFT i druge DSP algoritme.

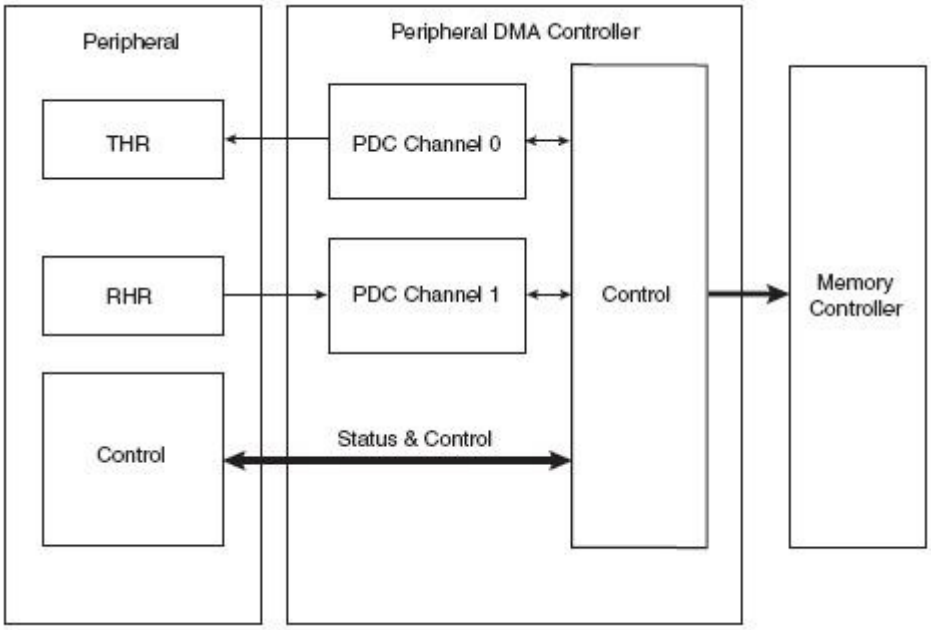
ARM7TDMI je procesor bez ***F****loating* ***P****oint* ***U****nit* (**FPU**) jedinice. Sve zadaće koje uključuju brojeve s decimalnom točkom moraju se softverski emulirati. Ukratko, ARM7TDMI je više nego znatno sporiji procesor od DSP procesora ekvivalentnog radnog takta ukoliko bi se uspoređivale performanse u DSP aplikacijama.

Nakon sto se prikupi N uzoraka potrebnih za *prvi krug* FFT-a, procesor mora izračunati spektar prije nove serije od N uzoraka. Ukoliko to ne stigne na vrijeme, izračun nije u realnom vremenu.

Kako bi se izbjeglo prekidanje izvršavanja algoritma svaki put kad AD pretvornik uzme jedan uzorak, koristi se DMA sklop. Naime, ukoliko DMA sklop ne bi postojao ili se koristio, procesor bi bio primoran nakon svakih 1/44100 sekundi vremena prekinuti izvršavanje, ući u prekidnu rutinu, pročitati podatak i spremiti ga u memoriju. Sve te operacije, plus povratak iz prekidne rutine, koštaju vremena.

DMA sklop radi na način da bez *puno* uplitanja procesora sam prebacuje podatke unutar dostupnog adresnog prostora, tj. dvije memorije (može naravno i selit podatke unutar iste memorije).

U našem slučaju, jedna memorija je ona od AD pretvornika gdje se nalazi uzeti uzorak (SFR registar koji je *uklopljen* u adresni prostor procesora), a druga radna SDRAM memorija procesora.



Slika 27. Blok shema DMA sklopa – preuzeto iz [6]

Slika 21. prikazuje pojednostavljenu blok shemu DMA sklopa.

Svaki sklop periferije kod koje DMA ima smisla, povezan je s centralnim DMA kontrolerom pomoću dva kanala. Jedan služi za prijenos podataka iz općeg adresnog prostora u periferijski sklop (***T**ransmit*), a drugi za prijenos iz perifernog sklopa *u memoriju* (***R**eceive*). AD pretvornik zbog svoje naravi nema *transmit* kanal.

Kako bi DMA kontroler svaki uzeti uzorak sam pohranio u memoriji, potrebno ga je inicijalizirati na sljedeći način:

- Podesit ***R**eceive **P**ointer **R**egister* (**RPR**)
  - RPR je 32-bitni registar u kojeg je potrebno upisati adresu od koje će se dolazni podaci spremati u memoriju. Nakon svakog prenesenog podatka, pokazivač se uvećava za 1,2 ili 4 ovisno o tome kakvi se podaci prenose.
- *Podesit **R**eceive **C**ount **R**egister* (**RCR**)
  -programiranje ovog registra određuje broj podataka koje DMA mora prenijeti, prije nego što obavijesti procesor da je *gotov*. RCR je *obično* brojilo koje broji unazad (prema nuli). Kao postigne vrijednost nula, prijenos je gotov. Za ovaj projekt potrebno ga je podesiti na N, tj. broj uzoraka signala nad kojima se vrši FFT analiza.

- **RN**ext**PR** i **RN**ext**CR**
  -ovi registri imaju istu ulogu kao i njihovi imenjaci bez dodatka «*Next*». Razlika je u tome što se upisom u njih može započeti novi prijenos podataka, odmah nakon što je prošli završio i prije nego se obavijesti procesor.

Prijenos svih N uzoraka signala u memoriju odvija se uz pomoć DMA kontrolera uz minimalnu intervenciju procesora. Za vrijeme prijenosa podatka kojima upravlja DMA, procesor nema pristup sabirnicama.

Da bi se dogodio prijenos, DMA sklop mora dobiti dopuštenje od procesora. Logika koja određuje smije li DMA koristiti sabirnicu (arbitraža) je projektirana na način da to dopusti onda kad je sigurna (ili skoro sigurna) da u tom razdoblju procesor neće koristiti sabirnicu. Pošto DMA sklop prenosi jedan podatak svakih 1/44100 sekundi, gotovo je sigurno (ne može se sa sigurnošću tvrdit) da niti na jedan način ne usporava izvršavanje algoritma.

U nastavku je prikazan programski odsječak koji izvršava potrebnu inicijalizaciju DMA sklopa. Memorijska adresa na koju će se prenositi podaci, odabrana je tako da stane 1024 uzorka (maksimalan broj uzoraka za koji će FFT raditi) gledano od vrha memorije.

Za točno značenje korištenih **SFR** registara potrebno je proučiti [6], poglavlje 22.

```
#define ENABLE_RECEIVER  0x01
void PDC_Init()
{
       ADC_RPR=0x0020F800;
       ADC_RCR=NUM_FFT;
       ADC_RNPR=0;
       ADC_RNCR=0;
       ADC_PTCR=(ENABLE_RECEIVER << 0);
}
```

4.3 Opći oblik programa

Opći oblik programa (funkcije *main()*) koji sadrži sve do sada razrađene komponente prikazan je u nastavku.

```
#define  JEDAN_PROLAZ   0
#define  TRUE           1

void main(void)
{
      //Inicijalizacija svih potrebnih varijabli i alociranje memorije
      int …
      unsigned int …
      //Inicijalizacije periferije
      PDC_Init();
      ADC_Init();
      TC0_Init();

      while(TRUE)
      {
            //Cekaj da N uzoraka bude spremno
            while(!(ADC_SR & 0x40000));
            //FFT i ostale operacije
            //Reinicijaliziraj DMA sklop
            PDC_Init();
            while(JEDAN_PROLAZ);
      }
}
```

Prije bilo kakve obrade signala potrebno je inicijalizirati DMA kontroler, AD pretvornik, te brojilo na samom početku programa. Prije toga je u memoriji osiguran prostor za pohranu prikupljenih podataka (uzoraka).

Glavni dio programa čini beskonačna petlja. Na samom početku, program čeka DMA kontroler (čiji je status integriran u *status* registar AD pretvornika) dok se ne prikupi svih N uzoraka signala.

Čim se uzorci prikupe, procesor ima zadaću maksimalnom brzinom proračunati spektar i pretvoriti ga u oblik pogodan za ispis na *Dot Matrix Display.* Kada se to dogodi i krene nova iteracija petlje, program mora biti u stanju čekanja novih uzoraka. Ukoliko to nije slučaj, uzorci su se prikupili prije nego je završila zadnja iteracija obrade spektra i sustav više ne radi u realnom vremenu.

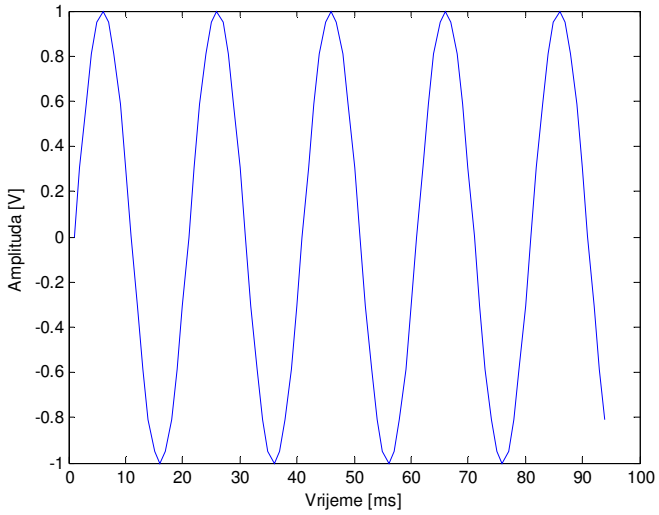
Konstantom *JEDAN_PROLAZ* moguće je radi svrhe testiranja zaglaviti program nakon jedne iteracije i pogledati dobiveni spektar.

U nastavku dokumenta opisane su same operacije koje pretvaraju ulazne uzorke u spektar pogodan za ispis na *display*.

4.4 Predobrada signala

Prije propuštanja uzoraka kroz FFT algoritam, potrebno je uzorke pretvoriti u oblik pogodan za daljnju obradu. Pri tome se prvotno misli na otklanjanje nedostataka koje unosi unipolaran AD pretvornik.

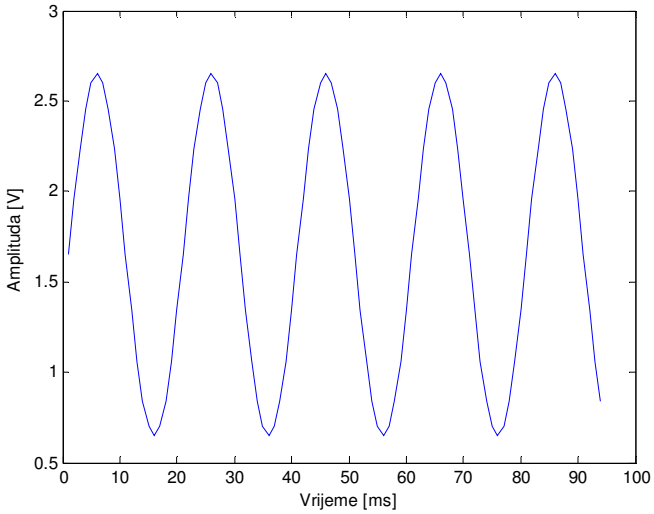
Neka je y(t) analogan signal koji se želi uzorkovati.



Slika 28. Analogan signal na ulazu

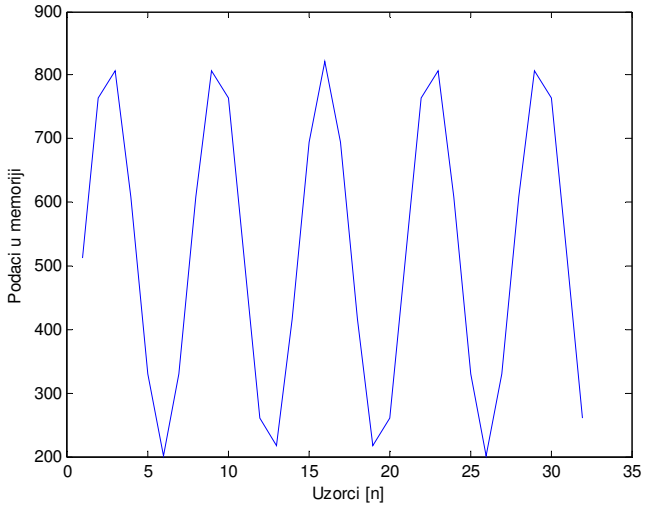
Na slici je prikazan sinusni signal amplitude 1. Trenutno nije bitna informacija o frekvenciji sinusa.

Zbog potrebe podešavanja statičke radne točke ulaza na pola referentnog napona, AD pretvornik osim sinusa vidi i DC komponentu jačine upravo V_{ref}/2. Takav je signal prikazan na sljedećoj slici.



Slika 28. Utjecaj radne točke na oblik signala

Nakon što se signal sa slike 28 uzorkuje 10-bitnim AD pretvornikom, koji se nalazi u AT91SAM7X256 mikrokontroleru, on poprima sljedeći oblik.



Slika 29. Signal nakon uzorkovanja

Oblik signala je dosta iskrivljen zbog konačne frekvencije uzorkovanja. AD pretvornik skalira ulazni signal iz intervala realnih brojeva 0 do 3.3V u interval cijelih brojeva između 0 i $2^{10}-1$. Pri tome je faktor pretvorbe jednak

$$\frac{2^{10}}{A}$$

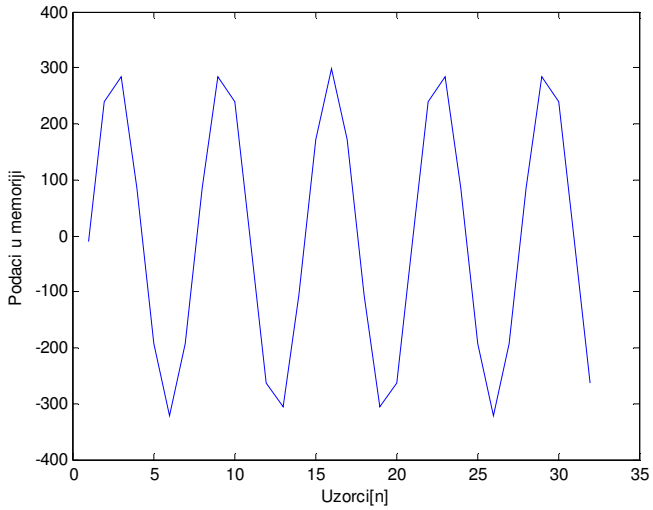
Nazovimo taj faktor *faktorom pretvorbe AD pretvornika,* pošto je on bitan ukoliko se želi dobiti točan spektar sa ispravno skaliranom amplitudnom osi.

Umjetno dodana DC komponenta signala je nepoželjna. Prva zadaća predobrade signala je skalirati signal tako da mu srednja vrijednost bude jednaka nuli. Ta će operacija isključivo pozitivne ulazne uzorke, koji se mogu prikazati *unsigned* tipom podataka, pretvoriti u *signed,* tj. brojeve s predznakom koji digitalna računala redovito prikazuju u formatu dvojnog komplementa (2'k prikaz).

Ilustrativan odsječak programskog kôda koji to izvršava prikazan je u nastavku.

```
//Odbijanje DC offseta i pretvorba u 2'k
for(i=0; i<NUM_FFT; i++)
      zbrojUlaza+=ReArray[i];
srednjaUlaza=zbrojUlaza/NUM_FFT;
for(i=0; i<NUM_FFT; i++)
      ReArray[i]-=srednjaUlaza;
```

U programskom odsječku, ReArray je pokazivač na dio memorije koji sadrži uzorke signala. Nakon opisane obrade, pohranjeni signal u memoriji procesora poprima oblik prikazan na slici 30.



Slika 30. Signal nakon odbijene DC komponente

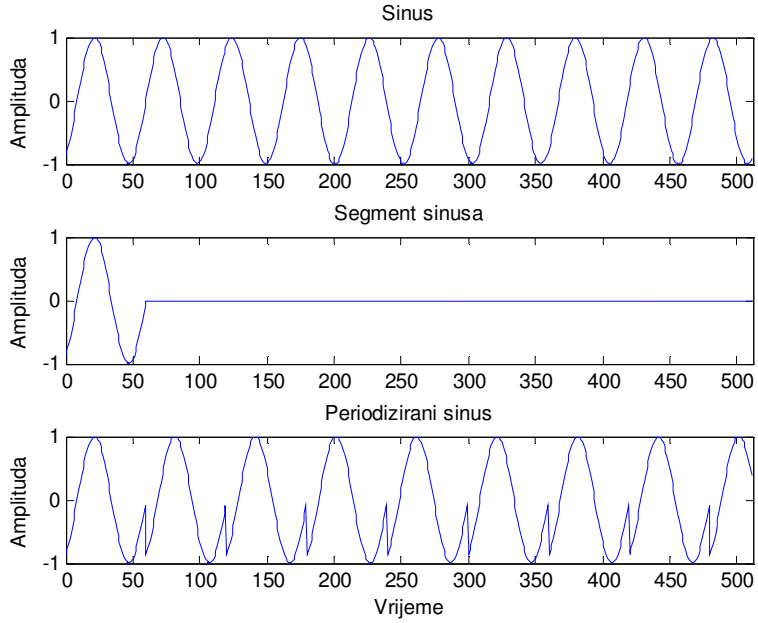
Tek je signal prikazan na slici 30. spreman za daljnju obradu.

4.5 Vremenski otvor

Da bi izračun spektra bio moguć koristeći **FFT** transformaciju, postavlja se uvjet da ulazni signal (niz uzoraka) bude periodičan. O razlozima postojanja ovog uvjeta može se pročitati u poglavlju 2.3.5.

Audio signal, kao i većina drugih, su klasični primjeri neperiodičnih signala. Čak i u slučaju da je ulazni signal čisti (periodičan) sinus, javlja se problem zbog **ograničenog vremena** tokom kojeg promatramo signal (uzimamo uzorke). Kada se signal otipka s N uzoraka, nakon vremena N*1/ f_s , ne može se znati kako se signal ponašao. Ponašanje je potrebno pretpostaviti, a Fourierova transformacija pretpostavlja da se signal periodično ponavlja upravo s periodom od tih N uzoraka.

Ukoliko smo bili te sreće da se broj uzoraka točno poklopio na način da je obuhvatio cijeli broj perioda signala, neće biti nikakvih problema. No više je vjerojatna suprotna situacija, kad N uzoraka ne sadrži cijeli broj perioda. Ovaj problem ilustrira sljedeća slika.



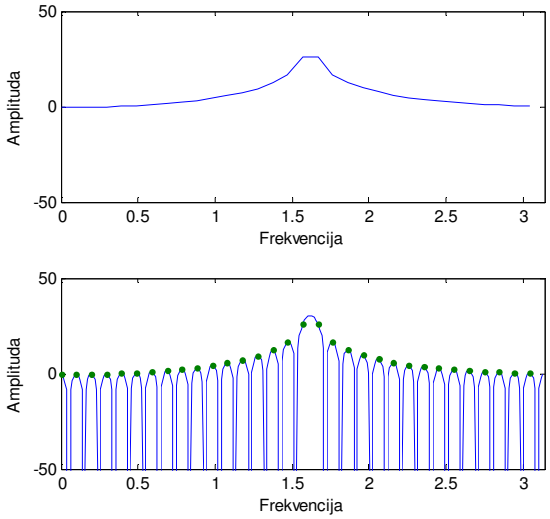
Slika 31. Problem ograničenog vremena uzorkovanja

Na prvoj slici prikazan je sinus jedinične amplitude i beskonačnog trajanja s *čistim* spektrom (može se zamisliti da vrijeme počinje i završava u $-\infty$ tj. $+\infty$). Digitalni sustav koji uzorkuje signal, snimio je u svoju memoriju samo jedan komadić sinusnog signala koji **ne** sadrži cijeli broj perioda.

Fourierova transformacija (bilo FFT ili DFT) pretpostavlja da su uzeti uzorci periodični, kao što je prikazano na podslici 3. Vrlo su uočljivi *teški* diskontinuiteti u signalu, koji će rezultirati vrlo zamrljanim spektrom, jer će se i oni pokušati aproksimirati odgovarajućim harmonicima.

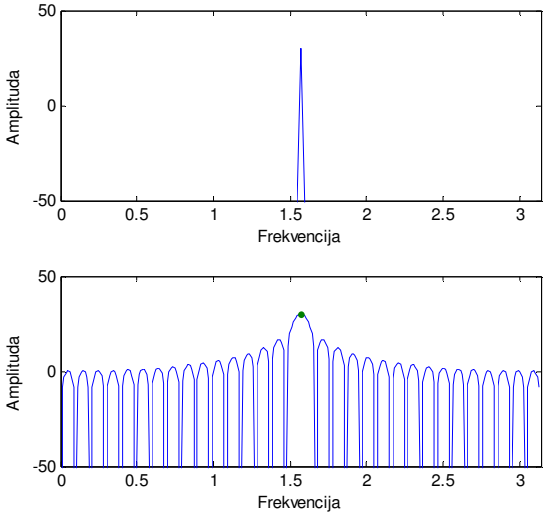
Ako se pak radi o signalu koji je po svojoj naravni neperiodičan (npr. audio signal), bez poduzimanja određenih mjera rezultat će biti identičan, tj. pogrešan, kako god da se uzorci poklopili (makar je s matematičkog gledišta Fourierove transformacije rezultat potpuno ispravan).

Izračun spektra neperiodičnog signala FFT-om podložan je pogrešci koja se naziva *spektralno curenje (*eng*.* ***spectral leakage****).* Ovaj efekt se jako lijepo vidi na sljedećoj slici i rezultat je ograničenog vremena snimanja (uzorkovanja) signala u slučaju periodičnih signala, tj. nedostatku periodičnosti u slučaju neperiodičnih signala. Na prvoj podslici prikazan je izlaz iz FFT-a. Energija glavnog harmonika iscurila je u susjedne. Ukoliko pogledamo drugu podsliku s prikazom **DT**ime**FT**-a vidi se gdje padaju točke koje kao rezultat uzima FFT.



Slika 32. Spektralno curenje

Ukoliko se broj uzoraka poklopio s cijelim brojem perioda, točke na DTFT grafu padaju na pravo mjesto, tj. FFT vidi samo jedan harmonik velike energije. Glavni harmonik maksimalno je izražen (pune energije), a ostale točke propale su u *nul točke.*

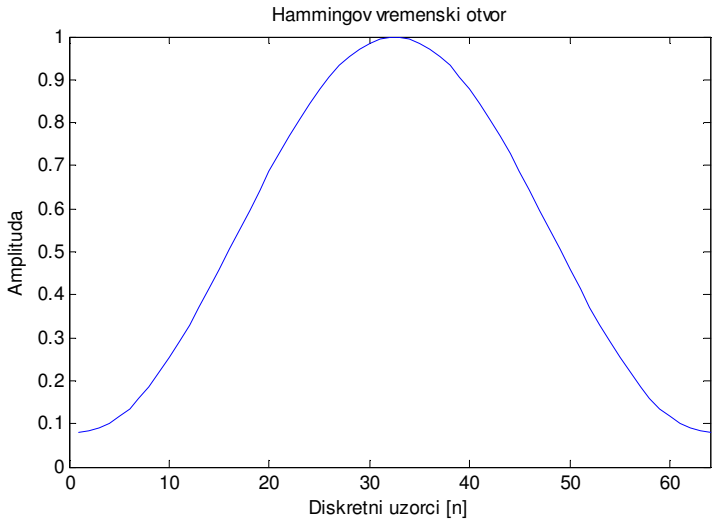


Slika 33. Nema curenja

Kako bi se ovaj problem maksimalno ublažio, na snimljeni komad signala primjenjuje se vremenski otvor ili kraće prozor (eng. *window*). On nije ništa više od matematičke funkcije koja množi točku po točku (uzorak po uzorak) prikupljeni signal.

Jasno je da je glavna namjena prozora na neki način periodizirati prikupljene uzorke, pa sam prozor da bi to ostvario mora gušiti signal na rubovima prema nuli. Prozora postoji mnogo, a jedan od njih po imenu *Hammingov* prozor, prikazan je na slici 33. Jednadžba koja opisuje ovaj prozor glasi:

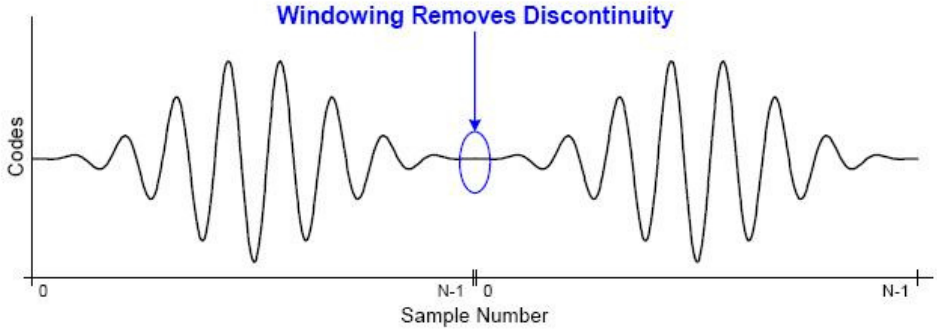
$$w(n) = 0.53836 - 0.46164\cos(\frac{2\pi}{N-1}n)$$



Slika 34. Hammingov prozor

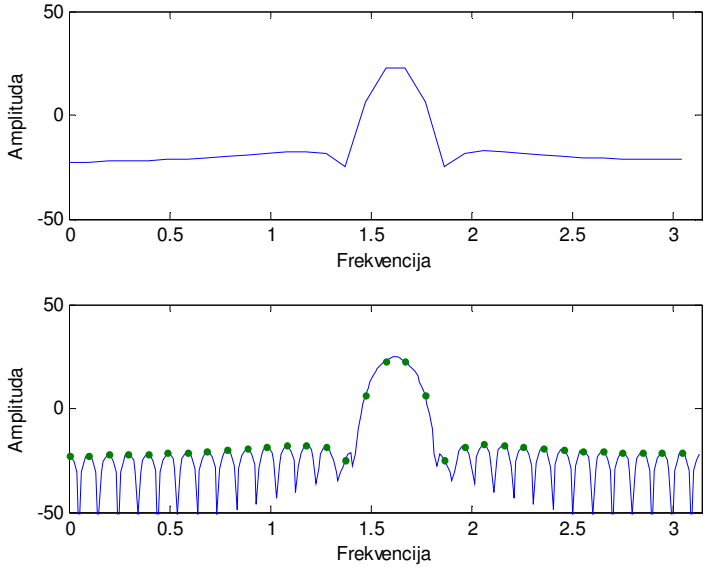
Različiti prozori odabiru se za različite namjene. Utjecaj prozora na spektar signala analizira se tako da se pogleda spektar samog prozora. Poznato je da umnožak signala u vremenskoj domeni rezultira konvolucijom u frekvencijskoj domeni. Prema tome, dobiveni spektar nije ništa više od konvolucije spektra ulaznog signala s spektrom prozora. Detaljnije razmatranje prozora izlazi iz okvira ovog završnog rada, pa se čitatelja koji želi znati više upućuje na [8].

Slika 34 demonstrira način rada prozora nad prikupljenim uzorcima i prikazuje izgled signala kako ga vidi FFT nakon primjene prozora.



Slika 35. Utjecaj prozora na signal

Ukoliko *Hammingov* prozor primijenimo na slučaj signala kada se broj uzoraka nije poklopio s cijelim brojem perioda, rezultat je sljedeći.



Slika 36. Primijenjen Hammingov prozor

Prvi susjedni harmonik spušten je za 43dB u odnosu na centralni, dok je širina centralnog 4 frekvencijska *bina* (na grafu se vide 4 točke). Ova dva parametra određuju karakteristiku prozora i razlikuju se od prozora do prozora.

ARM7TDMI je 32-bitni procesor bez FPU jedinice. Članovi prozora su realni brojevi u rasponu od 0 do 1. Kako bi se oni prikazali pomoću cijelih (*integer*) brojeva, softver mora na njih gledati kao na frakcije. 16 je odabran broj bita za prikaz frakcija. Npr. $1_{(10)}$ se transformira u $FFFF_{(16)}$, $0.5_{(10)}$ u $7FFF_{(16)}$ itd.

MATLAB kôd koji generira *Hammingov* prozor za N uzoraka, prikazan je u nastavku.

```
bw=16;
w=hamming(N)';
wc=round(w*2^(bw));
```

Generirani niz brojeva sprema se u memoriju kao ***L**ook **U**p **T**able* (**LUT**).

```
// Hamming Window
const unsigned int  WindowFunc[N/2] =
{
0x147A, 0x36F8 ……
};
```

Ključna riječ *const* je pred naredba prevodiocu da dotično polje spremi u ROM (*FLASH*) memoriju procesora, ali da ga kao takvog i koristi (jer spremiti ga mora u oba slučaja). U suprotnom, bi prevodilac automatski cijelo polje prebacio u RAM memoriju i time trošio vrijedne resurse.

Sama funkcija koja primjenjuje vremenski otvor je jednostavna i sastoji se od množenja istih indeksa uzoraka signala i prozora.

```
void Prozor (int ReArray[])
{
      int i;
      for(i=0; i< NUM_FFT/2; i++)
      {
            ReArray[i]*=WindowFunc[i];
            ReArray[NUM_FFT-i-1]*=WindowFunc[i];
      }
}
```

Kao i u slučaju predobrade signala, bitno je odrediti *faktor pretvorbe primjene prozora*. On iznosi

$$2^{16}\frac{\sum_{n=0}^{N-1} w(n)}{2}$$

Prvi član posljedica je prikaza realnih brojeva pomoću 16-bitnih frakcija. Drugi član je također konstantan i definira *gain* (pojačanje) samog prozora koje ovisi isključivo o tipu prozora i broju uzoraka. Sprema se u memoriju kao konstanta u obliku frakcije, a njeno određivanje lako obavi MATLAB.

Odabir korištenih prozora u ovom analizatoru sužen je na sljedećih 5.

- Bez prozora (odgovara pravokutnom eng. r*ectangular prozoru*)
- eng. *Triangle* (trokutasti) ili *Bartlettov* prozor
- *Julius von Hannov prozor*
- *Hammingov* prozor
- *Blackmanov* prozor

4.6 Povećanje preciznosti izračuna

Zbog činjenice da se svi realni brojevi prikazuju u obliku frakcija u registrima konačne veličine (32-bita), maksimalna točnost se postiže onda kada su registri najbolje iskorišteni, tj. kada se brojevi prikazuju kao 32-bitna frakcija.

Iz tog razloga, poželjno je sve podatke nakon primjene prozora posmaknut u lijevo za maksimalan mogući broj. To odgovara množenju s 2^{b_lijevo} . *B_lijevo* je konstanta koja određuje *faktor pretvorbe povećanja preciznosti*. Jednaka je broju mogućih posmaka najvećeg podatka, nakon množenja s prozorom. Kada se konstanta odredi, svi se podaci zajedno posmiču u lijevo, upravo za b_lijevo mjesta.

Programski kôd koji ovo izvodi vrlo je jednostavan i zbog toga nije naveden. Ukupan programski kôd nalazi se na CD-u priloženom uz završni rad.

4.7 FFT algoritam

Nakon što se izvršila predobrada podataka, primijenio vremenski otvor i povećala preciznost, podaci su spremni za izračun spektra FFT-om. Sukladno poglavlju 3.2 za izračun se koristi Radix-2 FFT algoritam.

Radix-2 algoritam zahtijeva indeks *bit reversing* koji je zadnja operacija nad uzorcima signala, prije same spektralne analize.

S obzirom da bi softversko okretanje svakog pojedinačnog indeksa predstavljalo osjetno opterećenje procesora, opet se koristi LUT tablica, kao i kod primjene vremenskog otvora u poglavlju 4.5.

U LUT tablici pohranjeni su *bit reversed* indeksi, koje program čita i vrši zamjenu. Zbog naravi samog okretanja indeksa, u memoriju je potrebno pohraniti samo pola indeksa, dok se ostali jednostavno proračunaju. Za detalje oko okretanja indeksa, čitatelj se upućuje na poglavlje 3.2. Programski odsječak koji preslaguje uzorke na taj način prikazan je u nastavku.

```
void Bit_Reverse(int BR_Array[])
{
      unsigned char swapA, swapB, sw_cnt;
      int TempStore;
      for (sw_cnt = 1; sw_cnt < NUM_FFT/2; sw_cnt++)
      {
            swapA = sw_cnt;
            swapB = BRTable[sw_cnt]*2;
            if (swapB > swapA)
            {
                  TempStore = BR_Array[swapA];
                  BR_Array[swapA] = BR_Array[swapB];
                  BR_Array[swapB] = TempStore;
            }
            swapA += NUM_FFT/2;
            swapB++;
            if (swapB > swapA)
            {
                  TempStore = BR_Array[swapA];
                  BR_Array[swapA] = BR_Array[swapB];
                  BR_Array[swapB] = TempStore;
            }
      }
}
```

Nakon što su uzorci ispravno poslagani, ulaze u sam FFT. FFT algoritam načelno se sastoji od 3 ugnježdene petlje. Cijeli programski kôd je prevelik da bi se uključio

u ovaj dokument. Kompletan programski kôd nalazi se na CD-u priloženom uz završni rad. Ovdje je samo kratko objašnjenje algoritma napisanog u C-u.

Radi lakšeg razumijevanja, čitatelj može pogledati sliku 19 koja predstavlja dijagram toka.

Sam FFT algoritam napisan za ovaj projekt, dodatno je optimiziran tako da predviđa posebne slučajeve *twiddle faktora,* te činjenicu da su u prvoj razini (eng. *stage*) sve imaginarne vrijednosti signala jednake nuli.

Zbog toga se početna petlja odnosi samo na prvi *stage (*nivo*)* FFT-a kada je argument jednak 0 (sinus je 0, a cosinus 1). Proračun *twiddle* faktora je u tom slučaju nepotreban. Uzima se u obzir, da su sve imaginarne vrijednosti nula, što dodatno ubrzava proračun prve razine. Ostale se razine izračunavaju pomoću tri petlje čiji je opći oblik prikazan u nastavku.

```
while(stage <= NUM_FFT/2)
{
	indexA=0;
	sin_index=0;
	for(g_cnt=0; g_cnt<group; g_cnt++)
	{
		for(s_cnt=0; s_cnt<stage; s_cnt++)
		{
			indexB = indexA + stage;
			//Ukoliko je moguće ubrzati izračun
			if(sin_index == 0) …
			elseif(sin_index == NUM_FFT/4)
			else
			{
			    //Ako nema prečaca računaj twiddle faktore
			}
			// Spremi rez. u memoriji i prati
			// prekoračenje opsega b_desno
			indexA++;
			sin_index+=group;
		}
		indexA = indexB + 1;
		sin_index = 0;
	}
	group /= 2;
	stage *= 2;
}
```

Tablica sinusa izvedena je u obliku LUT tablice i pohranjena u memoriju. Zbog mogućnosti određivanja sinusa u sva četiri kvadranta, poznavajući samo prvi, u memoriji je pohranjeno samo N/4 uzoraka sinusa.

Varijable *indexA* i *indexB* uvijek pokazuju na dva kompleksna broja koja se sljedeća obrađuju pomoću *butterfly* jednadžbi. W*hile* petlja određuje završetak programa, prateći trenutni razmak kompleksnih brojeva na ulazu u *butterfly*.

Dvije *for* petlje služe kako bi obavile sve *butterfly operacije* u jednoj razini prema slici 19.

Algoritam provjerava posebne slučajeve *twiddle faktora* koji ovise o trenutnom *sin_indeksu*. Indeks nula odgovara argumentu 0, pa nije potrebno računati *twiddle faktor,* jer *je sinus* jednak 0, a *coinus* 1. Sljedeće ubrzanje se postiže detektiranjem argumenta od $\pi/4$ gdje *sinus* iznosi 1, a *cosinus* 0.

FFT algoritam ima svojstvo pojačanja ulaznog skupa podataka. Maksimalno pojačanje (povećanje) brojeva na izlazu iz algoritma iznosi N/2 u slučaju realnih ulaza (AD pretvornik očitava smo realne vrijednosti ulaznog signala).

U poglavlju 4.6 obavljeno je povećanje preciznosti proračuna, tako da su podaci na ulazu u FFT već maksimalno popunili dozvoljena 32-bita. Bez ikakve dodatne intervencije očito je, da bi brojevi izašli iz opsega i proračun time bio netočan.

Zbog toga se prilikom spremanja podataka unaprijed predviđa kada će se dogoditi prekoračenje. Ukoliko bilo koji broj, koji se trenutno sprema u memoriju, prekorači polovicu vrijednosti opsega ($> 2^{30} - 1$ za brojeve s predznakom), program postavlja zastavicu koja signalizira sljedećoj razini da sve brojeve pročita posmaknute u desno za jedno mjesto (podijeljene sa 2). Time se nepovratno gubi *least significant bit* (LSB) podatka, no obzirom da se podaci namjerno (poglavlje 4.6) drže na vrhu registra, taj gubitak preciznosti se gotovo ne osjeti.

Ukupan broj posmaka u desno, akumulira se u brojaču *b_desno*. Time je određen *faktor pretvorbe* FFT algoritma koji iznosi

$$2^{-b\_desno}$$

Nakon što algoritam završi, spektar je pohranjen u memoriji u obliku kompleksnih brojeva. Brojevi su prikazani pomoću dva polja *integera ReArray* i *ImArray* veličine N.

4.7 Postobrada dobivenog spektra

Za analizator spektra audio signala potrebno je iz spektra izvući amplitude pojedinih harmonika, dok se faza zanemaruje. FFT na izlazu daje N kompleksnih brojeva, tj. N parova realnih i imaginarnih komponenti brojeva koji se u literaturi često nazivaju *frequency bins.* Svaki *bin* sadrži informaciju o amplitudi i fazi harmonika za određeni raspon frekvencija.

Kada uzmemo u obzir simetričnost spektra oko polovice frekvencije otipkavanja, daljnju obradu dobivenih podataka potrebno je vršiti na samo prvih N/2 rezultata. Druga polovica predstavlja tzv. negativne frekvencije i može se zanemariti.

Prva faza obrade rezultata uključuje izvlačenje amplitude iz pojedinih *binova*.

$$X(k) = \mathrm{Re} + j\,\mathrm{Im} = \mathrm{Re}\,Array[k] + \mathrm{Im}\,Array[k]$$

$$|X(k)| = \sqrt{\mathrm{Re}\,Array[k]^2 + \mathrm{Im}\,Array[k]^2}$$

Sam je izraz vrlo jednostavan, no nikako nam se ne sviđa ideja korištenja korijena, jer je njega teško implementirati procesorom bez FPU jedinice. Zbog toga se amplituda u prvoj fazi izračunava bez korijena, što se lako kompenzira u daljnjim fazama obrade, kao što će biti prikazano u nastavku.

```
for(i=0; i<NUM_FFT/2; i++)
{
      apsolutno[i].l = (long long)ReArray[i]*ReArray[i]
                     + (long long)ImArray[i]*ImArray[i];
}
```

Rezultat množenja garantirano prelazi opseg 32-bitnih brojeva i potrebno ga je spremati u 64-bita. To se postiže definiranjem tipa podatka *long long,* koji podatke tretira kao 64-bitne (*long* nije isto što i *long long*, ovo je osobina korištenog Keilovog C prevodioca).

Prije daljnjeg objašnjenja obrade bitno je definirati kako *binovi* odgovaraju stvarnim frekvencijama. Iz FFT izraza vidljivo je da k-ti izlaz odgovara argumentu

$$\frac{2\pi}{N}k$$

Pošto je valjani opseg argumenta $[-\pi,\pi]$, što prema *Shannonovom* teoremu odgovara frekvencijama $[-\frac{F_S}{2},\frac{F_S}{2}]$, kada umjesto π u izraz (gornji) ubacimo F_S/2, dobije se relacija koja povezuje indekse *binova* sa stvarnim frekvencijama.

$$\frac{k}{N}F_s$$

Da bi analizator spektra točno izračunao amplitude harmonika prisutne u ulaznom analognom signalu, potrebno je upotrijebiti sve faktore pretvorbe uvedene u prošlim poglavljima. Kao mali podsjetnik oni su popisani u nastavku.

- Analogan se signal sprema u memoriju mikrokontrolera s faktorom pretvorbe AD pretvornika koji iznosi $\frac{2^{10}}{A}$, gdje je 10 razlučivost pretvornika, a konstanta A iznos referentnog napona od 3.3V
- Faktor pretvorbe prozora iznosi $2^{16}\frac{\sum_{n=0}^{N-1}w(n)}{2}$. Broj bita odabran za prikaz frakcije realnih brojeva je 16, dok suma vrijednosti prozora podijeljena s 2 predstavlja konstantu specifičnu za broj uzoraka signala N (toliko iznosi i broj diskretnih uzoraka prozora) i tip vremenskog otvora (prozora).
- Povećanje preciznosti izračuna nakon primjene prozora unosi faktor pretvorbe od 2^{b_lijevo}.
- FFT algoritam ima svoj faktor pretvorbe koji iznosi 2^{-b_desno}

Ako se na ulaz priključi generator sinusnog signala amplitude 1V, željeni izlaz iz analizatora u frekvencijskom *binu* s indeksom koji odgovara frekvenciji analognog sinusa je amplitude 0dB.

Kako bi se to dobilo, potrebno je prebaciti amplitudni spektar u logaritamsku skalu.

$$X(k) = 20\log_{10}(|X(k)|) \quad [dB]$$

Gornji izraz sada proširujemo sa svim faktorima pretvorbe kako bi rezultat točno odgovarao ulaznom signalu.

$$X(k) = 20\log_{10}\left(\frac{2^{10}2^{16}2^{b\_lijevo}2^{-b\_desno}}{A}\frac{\sum_{n=0}^{N-1} w(n)}{2}\right) - \frac{1}{2}20\log_{10}(\mathrm{Re}\,Array[k]^2 + \mathrm{Im}\,Array[k]^2)$$

$$= 20\log_{10}(2^{10+16+b\_lijevo-b\_desno}) - \frac{1}{2}20\log_{10}(|X(k)|) + 20\log_{10}\left(\frac{\sum_{n=0}^{N-1} w(n)}{2A}\right)$$

$$= \left[\log_2(2^{26+b\_lijevo-b\_desno}) - \frac{1}{2}\log_2(|X(k)|)\right]\frac{20}{\log_2 10} + const$$

$$= \left[(26 + b\_lijevo - b\_desno) - \frac{1}{2}\log_2(|X(k)|)\right]\frac{20}{\log_2 10} + const$$

Jednom polovicom (0.5) ispred amplitudnog spektra nadomješten je nedostatak korijena. Skroz lijevi član je cjelobrojan i lako ga je izračunati. Faktor koji množi zagradu približno iznosi 6.02059 i sprema se u memoriju kao 64-bitna konstanta (cijeli broj + frakcija).

Konstanta s desne strane neovisna je o tijeku izračuna spektra, jer ovisi samo o broju uzoraka N i tipu prozora. Zbog toga se i ona sprema u memoriju kao 64-bitna konstanta.

Jedini problem u izračunu predstavlja određivanje logaritma amplitudnog spektra. Prvi korak pojednostavljenja prebacio je izračun logaritma iz baze 10, u bazu 2. Posmak registra odgovara množenju, tj. dijeljenju s 2 što omogućuje digitalnim računalima lakši izračun logaritma baze 2.

Uzmimo za primjer da želimo izračunati točan logaritam od broja 4660 ($0001234_{(16)}$). Za praćenje rezultata odmah definiramo rezultat koji približno iznosi $12.18611_{(10)}$.

Broj iz primjera možemo zapisati kao $x = m2^{(32-b\_posmaka)}$.

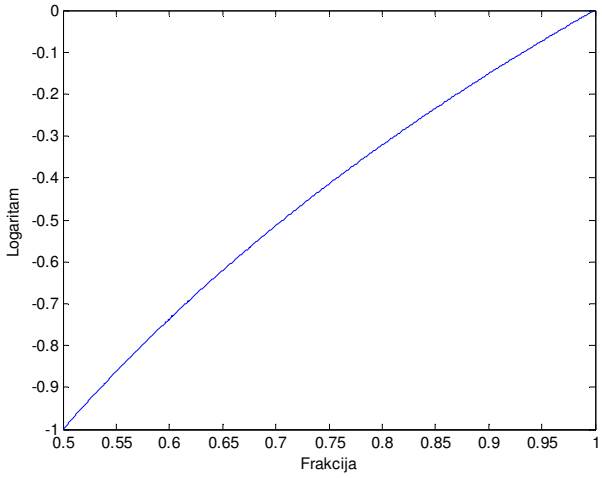
Brojanjem posmaka prvotnog broja u lijevo, sve dok se ne popuni maksimalan opseg registra, tj. bit najveće težine postane jedan, dolazimo do konstante **b_posmaka** koja za ovaj primjer iznosi 19.

Za novonastali broj **m** (*unsigned integer* jer je amplituda pozitivna) može se tvrditi da se nalazi negdje između $[0.5,1](2^{32}-1)$. To nam omogućuje promatranje tog broja kao frakciju veličine od 0.5 do 1. Upravo iz razloga što veliki *integer* broj promatramo kao mali realan broj, moramo na konstantu *b_posmaka* gledati kao da djeluje na suprotnu stranu registra (da dijeli broj, a ne množi) i zbog toga se u potenciji broja 2 nalazi (32-b_posmaka).

Primijenimo sada logaritam na novi izraz.

$$\log_2(x) = \log_2(m2^{(32-b\_posmaka)})$$
$$= \log_2(m) + (32 - b\_posmaka)$$

Jedino što preostaje je izračunati logaritam frakcije koja se može nalaziti u intervalu od 0.5 do 1. Vrijednost logaritma na tom intervalu prikazana je na sljedećoj slici.



Slika 37. Logaritam po bazi 2 u intervalu [0.5,1]

Krivulja logaritma na intervalu [0.5,1] aproksimira se polinomom drugog stupnja. Postoji više mogućih polinoma koji dobro aproksimiraju krivulju. Odabiremo uvjete:

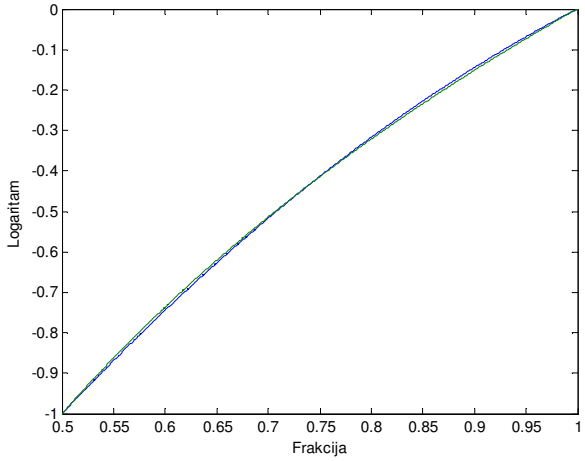
- parabola (polinom drugog stupnja) prolazi kroz rubne točke [(0.5,-1),(1,0)]
- ostale točke polinoma po metodi najmanjih kvadrata minimalno odstupaju

Proračun daje polinom sa sljedećim koeficijentima.

$$\log_2(m) \approx (-36 + \frac{24}{\ln(2)})m^2 + (56 - \frac{36}{\ln(2)})m + (-20 + \frac{12}{\ln(2)}) \qquad \forall m \in [0.5, 1]$$

Provjerimo sada odgovara li rezultat.

- b_posmaka iznosi 19
- frakcija m iznosi $\frac{4660 \cdot 2^{b\_posmaka}}{2^{32}-1} = 0.568847656$
- $\log_2(m)$ izračunat gornjim polinomom iznosi -0.821479953
- $\log_2(x) = \log_2(m) + (32-19) = 12.18611$



Slika 38. $\log_2(m)$

Kako bi što više ubrzali izračun polinoma koji ima sve realne koeficijente, izraz se može preurediti tako da je pri izračunu potrebno samo jedno množenje realnim brojem, dok se sve ostale operacije izvršavaju nad cijelim brojevima.

$$\log_2(m) \approx (-36m^2 + 56m - 20) + \frac{1}{\ln(2)}(24m^2 - 36m + 12)$$
$$\approx f_1(m) + const.f_2(m)$$

Zasebno se odrede rješenja dvaju odvojenih polinoma, gdje se samo drugi množi s realnom konstantom prije zbrajanja rezultata.

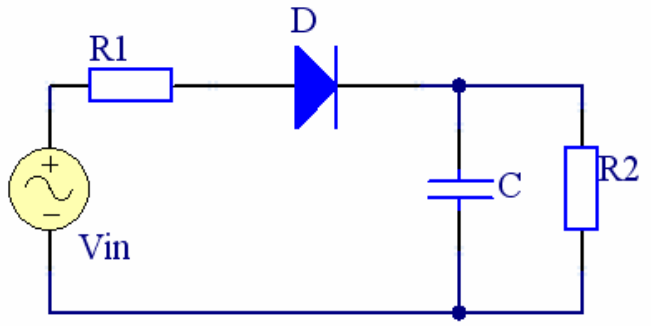
Postupak se ponavlja za svih N/2 dijelova spektra.

4.8 Usporenje prikaza

Audio analizator spektra mora izgledati atraktivno. Direktan prikaz spektra u decibelima izračunat u prošlom poglavlju ne bi dobro dočarao dinamiku ulaznog signala.

Kako bi se to izbjeglo, koristi se metoda eksponencijalne modifikacije pojedinih spektralnih komponenti koja sadrži dvije vremenske konstante. Jedna konstanta definira brzinu punjenja stupca spektra (na *Dot Matrix Displyu*), dok druga brzinu pražnjenja. Obično je na komercijalnim analizatorima pražnjenje osjetno sporije od punjenja, kako bi se kratki visoki harmonici primijetili u prikazu.

Princip rada može se prikazati jednostavnim električnim krugom prvoga rada.



Slika 39. Modeliranje prikaza električnim krugom

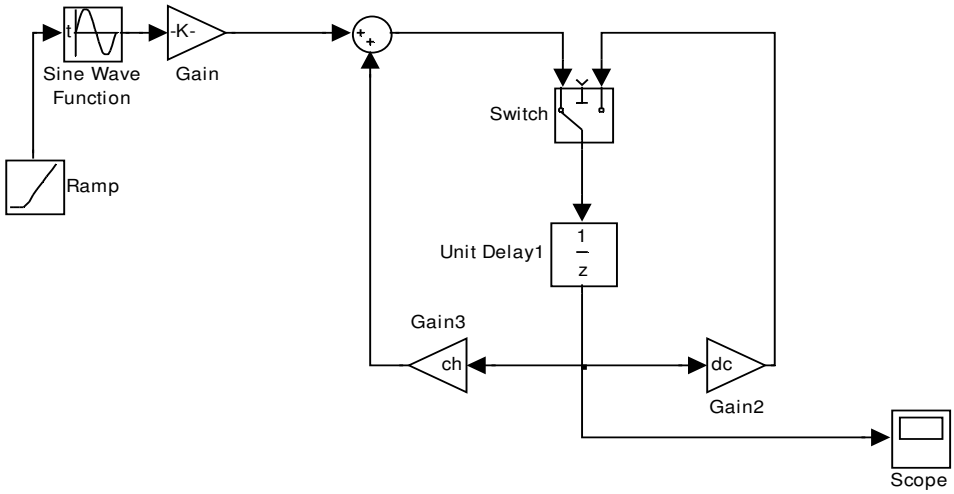
Neka je kondenzator nabijen na neku početnu vrijednost. Ako je amplituda ulaznog signala manja od napona na kondenzatoru, zbog diode, ulazni krug potpuno je odsječen od kondenzatora. Kondenzator će se prazniti vremenskom konstantom $\tau_{dc} = R_2 C$.

U trenutku kada amplituda ulaznog signala premaši trenutnu vrijednost napona na kondenzatoru, dioda počinje voditi i kondenzator se puni konstantom $\tau_{ch} = (R_1 \parallel R_2) C$.

Ovakav električni krug prvog reda lako se modelira na računalu.

Pri tome trenutni napon na kondenzatoru odgovara trenutnoj visini stupca spektra na *displayu* (koja ovisi o svim prošlim amplitudama spektra). Ulazni signal je spektar dobiven u novoj iteraciji analize spektra. Za svaku pojedinu spektralnu komponentu koja je veća od trenutne, sklopka određuje punjenje stupca

pojačanjem A_{CH} (od eng. *charge*). U suprotnome, stupac spektra se prazni s konstantom A_{DC} (od eng. *discharge*).



Slika 40. Računali model

Poznavajući frekvenciju osvježavanja spektra, moguće je odrediti pojačanja punjenja i pražnjenja u sekundama. Vremenska konstanta τ definirana je kao vrijeme potrebno da se kondenzator, s početne vrijednosti, isprazni na 36.78% što odgovara 1/e. Prema tome, možemo pisati

$$(A_{dc})^M = \frac{1}{e}$$
$$M \ln(A_{dc}) = -1$$
$$A_{dc} = e^{-\frac{1}{M}}$$

gdje M predstavlja broj potrebnih iteracija računalnog modela, a A_{dc} vremensku konstantu (tj. faktor pojačanja) pražnjenja.

Ako analizator radi u realnom vremenu uz N-point FFT i frekvenciju uzorkovanja f_S, frekvencija osvježavanja iznosi $f_r = \frac{f_S}{N_{FFT}}$.

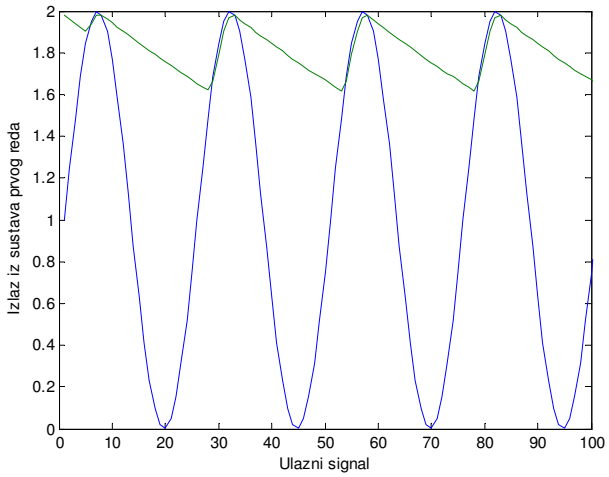
Želimo odrediti faktor pojačanja preko vremenske konstante u sekundama. $\tau[s] = \frac{1}{f_r} M = 1$ jer je M upravo recipročna vrijednost o frekvencije osvježavanja.

Uvrtimo sada izraz frekvencije osvježavanja f_r što dovodi do izraza za broj osvježavanja izražen preko vremenske konstante.

$$M = \frac{f_S}{N_{FFT}}\tau$$

I konačno, kada se gornji izraz unese u faktor pojačanja, dobiva se

$$A_{dc} = e^{-\frac{1}{\frac{f_s}{N_{FFT}}\tau}}$$



Slika 40. Odziv na sinus

Slika 40. prikazuje odziv sustava na sinusnu pobudu.

Dot Matrix Display

5.1 Uvod

Prikaznici temeljeni na matrici LED dioda danas nalaze široku primjenu. Razlog tome leži u niskoj cijeni samog ekrana u odnosu na njegovu veličinu, tj. površinu aktivnog dijela za prikaz slike.

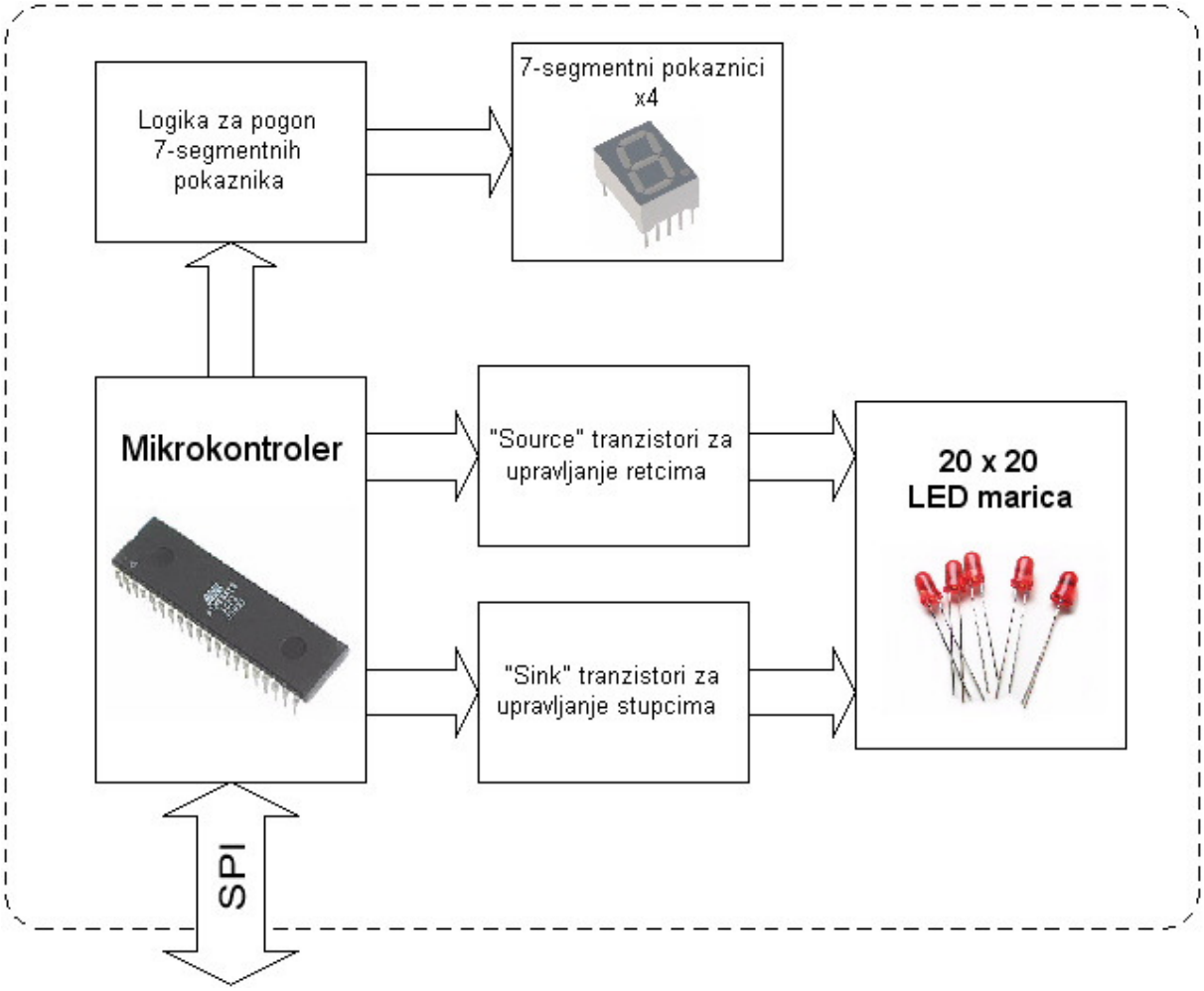
Glavna karakteristika im je vrlo mala rezolucija prikaza koja je određena veličinom matrice svjetlosnih dioda. Diode mogu biti jednobojne ili višebojne ovisno o namjeni ekrana.

Jednobojni ekrani se najčešće koriste za prikaz plivajućeg teksta i matrica im je duguljastog oblika. Višebojni ekrani služe za prikaz raznih animacija, najčešće reklama. Kvaliteta njihove izrade je visoka pa se promatraču iz daljine može činiti da gleda u LCD ekran.

Treba spomenuti i nisku potrošnju energije za što je zaslužan vremenski multipleks objašnjen u nastavku.

Za potrebe ovog završnog rada odlučeno je izraditi jednobojni *Dot Matrix Display* veličine 20x20 opisan u nastavku. Njegova je glavna namjena prikaz spektra audio signala, no može poslužiti i u mnogim budućim projektima kao ekran opće namjene.

5.2 Blok Shema

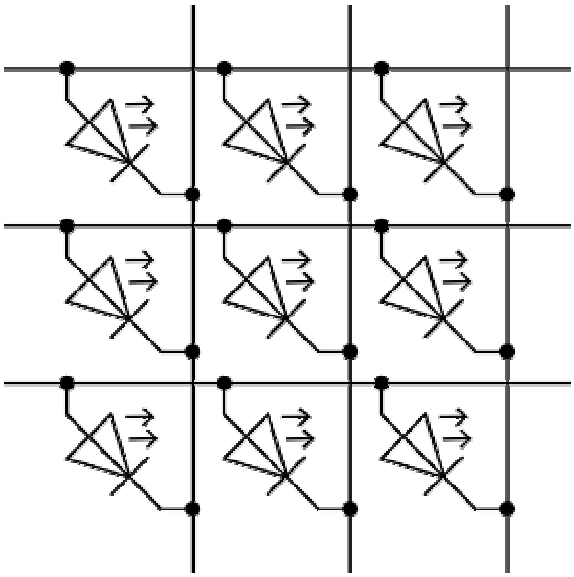


Slika 41. Blok shema Dot Matrix Display-a

5.3 Opis rada

5.3.1 Temeljna ideja

LED diode su organizirane u pravilnu matricu veličine 20 redaka i 20 stupaca.



Slika 42. LED matrica

Svaki redak spaja zajedno anode dioda u tom retku, a svaki stupac katode kao što je prikazano na slici iznad. Da bi se određena dioda upalila, potrebno je osigurati struju kroz tu diodu u propusnom smjeru. To znači da anoda (redak) mora biti pritegnuta visoko, a katoda (stupac) nisko, naravno, uz ograničenje struje.

Ovakav spoj dioda je idealan za rad ekrana u vremenskom multipleksu. Vremenski multipleks je ostvaren tako da u jednom trenutku maksimalno jedan stupac bude pritegnut nisko. Da bi se diode u tom stupcu upalile, potrebno je pomoću strujnog izvora, potjerati struju u retke dioda koje želimo upaliti.

Nakon toga, trenutno aktivan stupac se deaktivira i sljedeći postaje aktivan (pritegnut nisko). Opet se izborom redaka koji će provoditi (eng. *source*) struju određuje koje će se diode u tom stupcu upaliti.

Pošto *display* ima 20 stupaca, potrebno je gornjim postupkom proći kroz svih 20, ne bi li se prikazala željena slika. Brzina kojom se prolazi kroz stupce (i istovremeno pale određene diode u redcima) mora biti dovoljno velika da zavara ljudsko oko. Zavaravanje se temelji na tromosti oka, kojemu se čini da su sve

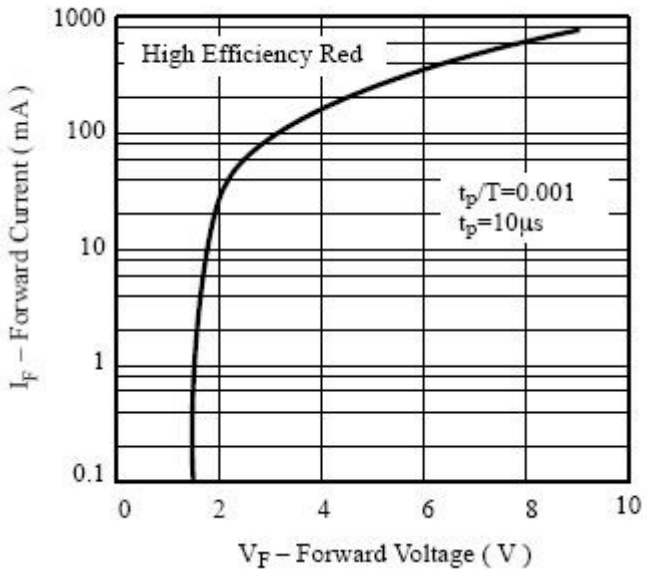
diode upaljene istovremeno. Temeljem testiranja već gotovog ekrana zaključena je statičnost slike na frekvenciji osvježavanja od 100Hz. Na toj frekvenciji mora raditi svaka dioda. Da se to ostvari, frekvencija aktiviranja stupaca mora biti minimalno 2000Hz! (pošto imamo 20 stupaca)

Vrijeme tokom kojeg je jedan stupac aktivan (u jednom prolasku) iznosi 1/2000Hz, tj. 500µs. Ukupan ciklus osvježavanja traje 1/100Hz, tj. 10ms. Radni omjer diode iznosi 0.05.

Za kvalitetno projektiranje sklopovlja koje će na ovaj način pokretati ekran, potrebno je prvo pogledati neke karakteristike korištenih dioda, prikazane u nastavku.

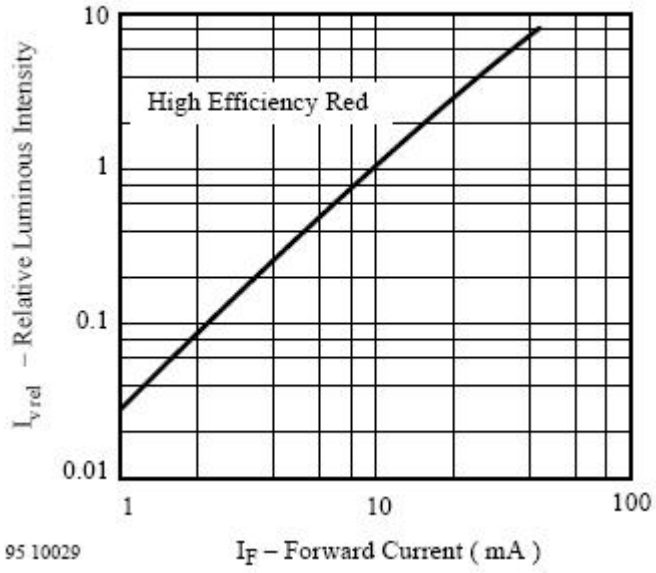
Tip diode: TLH6400, *Red diffused* 5mm kućište

Radni napon: 2V za struju od 20mA



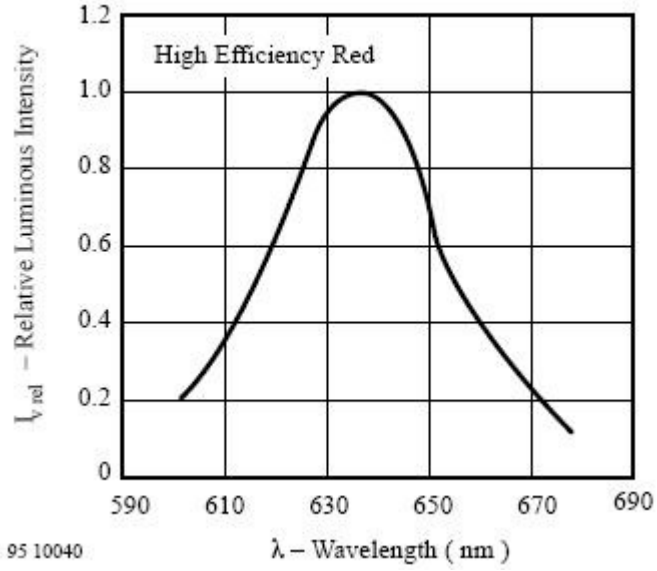
Slika 43. Naponsko- struja karakteristika diode

Radna struja: 20ak mA zbog malog radnoj omjera



Slika 44. Osvjetljenje u ovisnosti o radnoj struji

Valna duljina: oko 635nm (crvena boja)



Slika 45. Radni spektar diode

Pošto je okvirni način rada opisan i karakteristike korištenih dioda poznate, u nastavku se nalazi opis sklopovlja koji to ostvaruje.

5.3.2 Mikrokontroler

Jezgru sustava čini *Atmelov* 8-bitni mikrokontroler. On sadrži softver koji iscrtava sliku na ekran.

Shema spoja prikazana je na stranici 85 i 86. Da bi mikrokontroler radio, potrebno je, osim napajanja, dovesti jedino vanjski izvor stabilnog takta. To je učinjeno spajanjem kristala kvarca frekvencije 16MHz.

Treba napomenuti funkcionalnost korištenog mikrokontrolera koja omogućava korištenje internog RC oscilatora do 8MHz. On se ne koristi zbog premalog radnog takta za potrebe *Dot Matrix Display-a* ove veličine i nestabilne frekvencije osciliranja što je karakteristika svih RC oscilatora.

Atmel nudi više familija cijenom pristupačnih 8-bitnih mikrokontrolera. Za ovaj projekt izabran je **ATmega16** chip iz *AVR* familije. Karakteristike zbog kojih je baš ovaj chip izabran nalaze se u nastavku.

- **Chip:** Atmel AVR ATmega16
- **Kućište:** DIP40
- **Napajanje:** 5V DC @ 50mA maksimalno
- **Radni takt:** 16MHz maksimalno, većina instrukcija jednociklusne
- **I/O mogućnosti:** 4 porta, tj. 4x8 I/O priključaka.
- **Komunikacija:** SPI, TWI, USART

Najvažnija karakteristika je broj I/O priključaka koji iznosi 32. U slučaju korištenja chipa s malim brojem priključaka bilo bi neophodno raditi puno više proširenja portova nego što je to učinjeno (opisano u nastavku).

LED matrica ima 40 izvoda kojima treba upravljati (20 redaka + 20 stupaca). Pošto izabrani mikrokontroler ima *samo* 32, očito je da ne može direktno pokretati matricu.

Postoji jedna još veća prepreka od nedostatka priključaka. To je maksimalna struja koju izabrani mikrokontroler može dati (*source*), tj. primati (*sink*). Da bi odredili maksimalnu struju, razmotrimo na trenutak energetske karakteristike matrice u najgorem slučaju (sve diode upaljene).

Određeno je da će svaka dioda svijetliti strujom od 20mA. Obzirom da je u jednom trenutku maksimalno jedan stupac aktivan, to dovodi do 20 × 20mA = 400mA struje koju treba osigurati. U *datasheetu* mikrokontrolera nalaze se sljedeći podaci:

1) The sum of all IOL, for **all ports**, **should not** exceed **200 mA.**
2) *The sum of all IOL, for **one port**, **should not** exceed **100 mA.***

Ove vrijednosti se odnose za izvor (*source*) i ponor (*sink*) struje. Na jedan *port* spojeno je 8 dioda (8 × 20mA = 160mA, što nije dozvoljeno). Kada bi nekako i osigurali dovoljnu struju, mikrokontroler bi morao ponirati struju cijelog stupca (400mA) u jedan priključak što bi trenutno uništilo chip. Upravo se iz ovih razloga na blok shemi između mikrokontrolera i LED matrice nalaze dvije komponente opisane u nastavku.

5.3.3 *Sink* tranzistori za upravljanje stupcima

Između mikrokontrolera i stupaca matrice postavljeni su tranzistori tako da imaju mogućnost poniranja struje, tj. pritezanje stupca nisko. Na baze tranzistora spojeni su bazni otpornici, radi ograničenja struje kroz pn spoj baza-emiter. Shema spoja nalazi se na stranici 88.

Zbog maksimalne struje od oko 400mA, trebalo je odabrati pogodan tranzistor koji u statičkim uvjetima može podnijeti tu struju. Nakon pretrage kataloga domaćih distributera izabran je BC639. BC639 je NPN tranzistor s dozvoljenom strujom kolektora do 1A.

Da bi se odredio iznos otpora, potrebno je znati koliko iznosi faktor statičnog strujnog pojačanja. *Datasheet* dotičnog tranzistora navodi vrijednost 40 kao najgori mogući slučaj pojačanja. Maksimalna struja kolektora (stupca) iznosi otprilike 400mA . Za dovođenje tranzistora u zasićenje mora vrijediti relacija

$$I_B > \frac{I_C}{\beta}$$

tj. struja baze mora biti veća od 10mA. Treba napomenuti da je ovo apsolutno najgori slučaj pojačanja. U normalnim radim uvjetima pojačanje se kreće oko 100. Maksimalni iznos otpora određuje sljedeća relacija koja se izvodi iz sheme spoja.

$$R = \frac{5 - U_{BE}}{I_B} = 430\Omega$$

Pri tome je pretpostavljena vrijednost izlaza iz sklopa, koji pokreće bazu, od 5V. U sklop su ugrađeni otpornici od 332Ω isključivo iz razloga što su oni u tom trenutku bili dostupni.

Umjesto upravljanja stupcima direktno preko mikrokontrolerovih I/O priključaka, to čine tri dekadska brojača tipa 4017. Njima upravlja mikrokontroler koristeći samo 4 priključka. Treba razlikovati binarne i dekadske brojače. Dekadski brojač na svaki impuls takta postavlja sljedeći (0->1->2..) izlaz aktivan (visoko), dok binarni npr. nakon 7 impulsa postavlja tri svoja izlaza aktivno (u obliku binarnog broja).

Svaki od tri dekadska brojača ima 10 izlaza. Obzirom da je izlaz nula (0) aktivna pod *resetom*, ona se ne koristi pa ostaje 9. Iz tog razloga se koristi tri brojača kako bi svojim izlazima pokrili svih 20 stupaca.

Mikrokontroler preko jednog priključka daje signal takta, a preko još tri određuje koji će brojači biti pod *resetom*, tj. koji će brojati.

U programu su svi brojači početno pod *resetom*. Program aktivira prvi brojač, dok ostale i dalje drži pod *resetom*. U tom su trenutku svi stupci neaktivni, jer su svi brojači na nuli (izlaz 0 se ne koristi). Kada program pošalje impuls takta, prvi brojač prelazi u stanje «1», dok ostali ne reagiraju. Samim time prvi stupac postaje aktivan. Nakon sljedećeg impulsa, sljedeći izlaz postaje aktivan, jer je brojač u stanju 2. Taj postupak se ponavlja sve do stanja 9, u kojemu prvi brojač prelazi u stanje čekanja (*reset*), a program aktivira sljedeći brojač.

Ovaj sklop omogućuje mikrokontroler u kontrolu velikih struja uz malen broj I/O priključaka.

5.3.4. *Source* tranzistori za upravljanje redcima

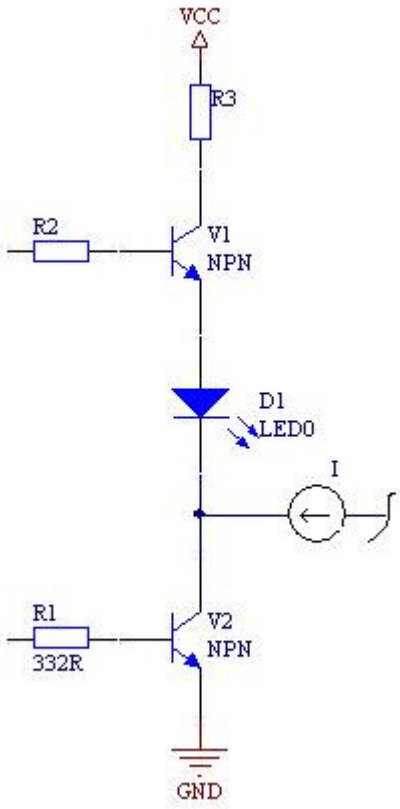
Kada gore opisan sklop aktivira neki od stupaca, program mora trenutno upaliti sve diode koje u tom retku čine konačnu sliku.

Da bi se diode zadovoljavajuće upalile, potrebno je osigurati struju kroz svaku od njih od oko 20mA. Kako je već opisano u poglavlju 2.3.2, sam mikrokontroler ne

može osigurati struju koja bi upalila više od par dioda. Da bi se dobila mogućnost paljenja svih 20 dioda istovremeno, između mikrokontrolera i redaka matrice spojeni su tranzistori.

Tranzistori su spojeni na način da omoguće izvor (eng. *source*) struje za pojedini redak. Shema spoja prikazana je na stranici 87.

Da bi odredili iznose otpornika, razmotrimo sljedeću pojednostavljenu shemu upravljanja matricom.



Slika 46. Prikaz jednog stupca

V2 je tranzistor koji služi za aktiviranje (pritezanje) stupca nisko. V1 je source tranzistor. Strujni izvor predstavlja ostalih 19 dioda u stupcu koje nisu nacrtane i opterećuje kolektor V2 tranzistora s dodatnom strujom.

Otpornik R2 mora osigurati zasićenje tranzistora V1 kada program želi aktivirati redak.

Otpornik R3 određuje struju koju tranzistor propušta u diodu D1.

Da bi odredili iznose otpornika, potrebno je prvo odrediti potencijal emitera gornjeg tranzistora

$$U_E = U_{CEzas} + U_D$$

Iz *datasheeta* BC639 tranzistora očitava se napon zasićenja od oko 0.5V. U poglavlju 2.3.1 određen je napon diode od 2V. Iz toga slijedi da je potencijal emitera gornjeg tranzistora 2.5V.

Pošto tranzistor V1 mora osigurati struju samo za jednu diodu (koja iznosi 20mA), izabran je BC547C koji se odlikuje velikim strujnim pojačanjem. Strujno pojačanje iz *datasheeta* za BC547C iznosi minimalno 400.

Jednostavnim proračunom kao u poglavlju 2.3.3 dobiva se maksimalan iznos otpornika R2. Radi dodatne sigurnosti izabran je nešto manji otpornik vrijednosti $10\text{k}\Omega$.

Struja koja teče kroz diodu D1 približno je jednaka struji kroz otpornik R3.

$$I_{R3} = \frac{V_{cc} - U_E}{R3}$$

Iz toga slijedi da R3 iznosi 125Ω. U sklop su ugrađeni otpornici od 120Ω (metal film tipa).

Ovakav spoj tranzistora ima dvojaku ulogu. Prva je jednostavno reguliranje struje dioda pomoću jednog otpornika. Druga, još važnija, je mogućnost upravljanja diodama putem mikrokontrolera koji mora osigurati samo male bazne struje tranzistora.

5.3.5 7-segmentni pokaznici

Kako bi ekran dobio dodatnu mogućnost jednostavnog ispisa brojeva, odlučeno sam ugraditi četiri 7-segmentna pokaznika. Oni također rade u vremenskom multipleksu radi uštede priključaka mikrokontrolera i energije.

Shema spoja prikazana je na stranici 90.

Da bi se 4 pokaznika osvježavala u vremenskom multipleksu, potrebno je osigurati minimalno 7(segmenti)+4(katode)+1(decimalna točka) priključaka. Na upravljanje matrice je potrošeno

- 20 za upravljanje redcima
- 4 za upravljanje stupcima
- 4 za komunikaciju s drugim sustavom koji šalje sliku i programiranje (opisano u nastavku)

Od ukupnog broja priključaka ostala su 4 slobodna. Pošto je potrebno minimalno 12, odlučeno je koristiti nekoliko međusklopova čija je glavna zadaća smanjiti potreban broj priključaka.

Tri priključka upravljaju sklopom D8 (74HC595). 74HC595 je «*serial in-parallel and serial out*» posmačni registar. Program na SER priključak postavi prvi bit. Zatim okine SRCLK što uzrokuje posmak registra unutar chipa. Nakon što se svih 8 bitova na ovaj način upiše, da bi se podatak pojavio na izlazu sklopa, potrebno ga je upisati u izlazni registar jednim impulsom na RCLK liniji.

Ovaj sklop je uštedio 5 priključaka mikrokontrolera, no to i dalje nije dovoljno za upravljanje pokaznicima. Iz toga razloga je na izlaz 74HC595 sklopa spojen D9 (CD4511). CD4511 je vrhunski sklop za upravljanje 7-segmentnim pokaznicima. Radi se o jednostavnom dekoderu koji dekodira ulazne BCD signale, u signale za prikaz dotičnog BCD broja na pokazniku. Osim toga, izlazni stupanj mu je posebno dizajniran da direktno pokreće LED diode unutar pokaznika, što isključuje uporabu tranzistora. Time su ušteđena dodatna tri priključka (4 ulazna generira 7 izlaznih).

Preostalih 4 priključka posmačnog registra spojeno je na D10 (ULN2004A). Radi se o polju od 8 darlingtonovih tranzistora unutar DIP16 kućišta. Oni služe za pritezanje zajedničkih katoda jednog (određuje program) pokaznika nisko (u tom trenutku aktivnog).

Potrebno je ograničiti struju iz CD4511 koja teče kroz diode, što je učinjeno DIP16 poljem od 8 otpornika označenih kao R61.

Četvrti priključak mikrokontrolera (samo se tri koriste za pogon posmačnog registra, koji s svojih 8 izlaznih priključaka upravlja svim ostalim sklopovima) iskorišten je za upravljanje decimalnom točkom (dot).

5.3.6 Napajanje

Zbog velike potrošnje LED dioda, sklop napajanja je nešto složeniji od standardne izvedbe s jednim linearnim stabilizatorom. Shema spoja prikazana je na stranici 91.

Sklop se sastoji od tri linearna stabilizatora tipa 7805. Napajanje je projektirano na način da izdrži udarne struje koje mora osigurati u nekim trenutcima, kao i statičku disipaciju snage koja je posljedica pada napona na samim regulatorima.

Sam regulacijski krug podijeljen je na dva dijela. Jedan služi isključivo za napajanje LED matrice, dok drugi napaja digitalnu logiku uključujući mikrokontroler

i 4 7-segmentna pokaznika (obzirom da se oni napajaju direktno iz chipa CD4511). Glavni razlog ove podjele je osigurati digitalnim chipovima stabilan napon napajanja. Matrica dioda naglo mijenja potrošnju struje koja se kreće između 0 i 400mA, što bi unijelo smetnje (neželjene harmonike) u napajanje digitalnih chipova.

Struju LED matrice osiguravaju dva 7805 regulatora spojena u paralelu. Diode V41 i V42 omogućuju podjednako opterećenje svakog regulatora. Naponske izvore je zabranjeno spajati paralelno, osim u teoriji kada se pretpostavlja da je njihov iznos potpuno jednak. U praksi uvijek postoji blago odstupanje u naponu, koje ovaj sklop kompenzira diodama. Radna točka dioda automatski se namjesti tako da se izlazni naponi parova regulatora i dioda izjednače.

Diode spuštaju potencijal izlaza s 5 na 4.3V. Uloga diode V43 je kompenzacija tog gubitka. Ona podiže referentnu točku regulatora za 0.7V, pa je ukupan izlaz 4.3 + 0.7 = 5V.

Za napajanje digitalnih chipova paralelno ulazu spojen je treći 7805 regulator. Konektori X2 i X3 služe za jednostavno prekidanje strujnog kruga. Skidanjem kratkospojnika koji su postavljeni u normalnom radu, lako se multimetrom mjeri potrošnja struje digitalne logike, tj. matrice u [mA].

5.4 Upravljanje i komunikacija

Display sam po sebi ne može raditi ništa pametno. Potrebno je spojiti drugi (u daljem tekstu eng. *master*) sustav koji će slati instrukcije i podatke *displayu.*

Komunikacija između *master* sustava i *displaya* odvija se putem SPI sabirnice. SPI je odabran zbog svoje komercijaliziranosti i široke primjene, zbog čega nalazi mjesto u gotovo svakom boljem kontroleru. Odabir široko primjenjivane I2C sabirnice je izostao zbog dva glavna razloga

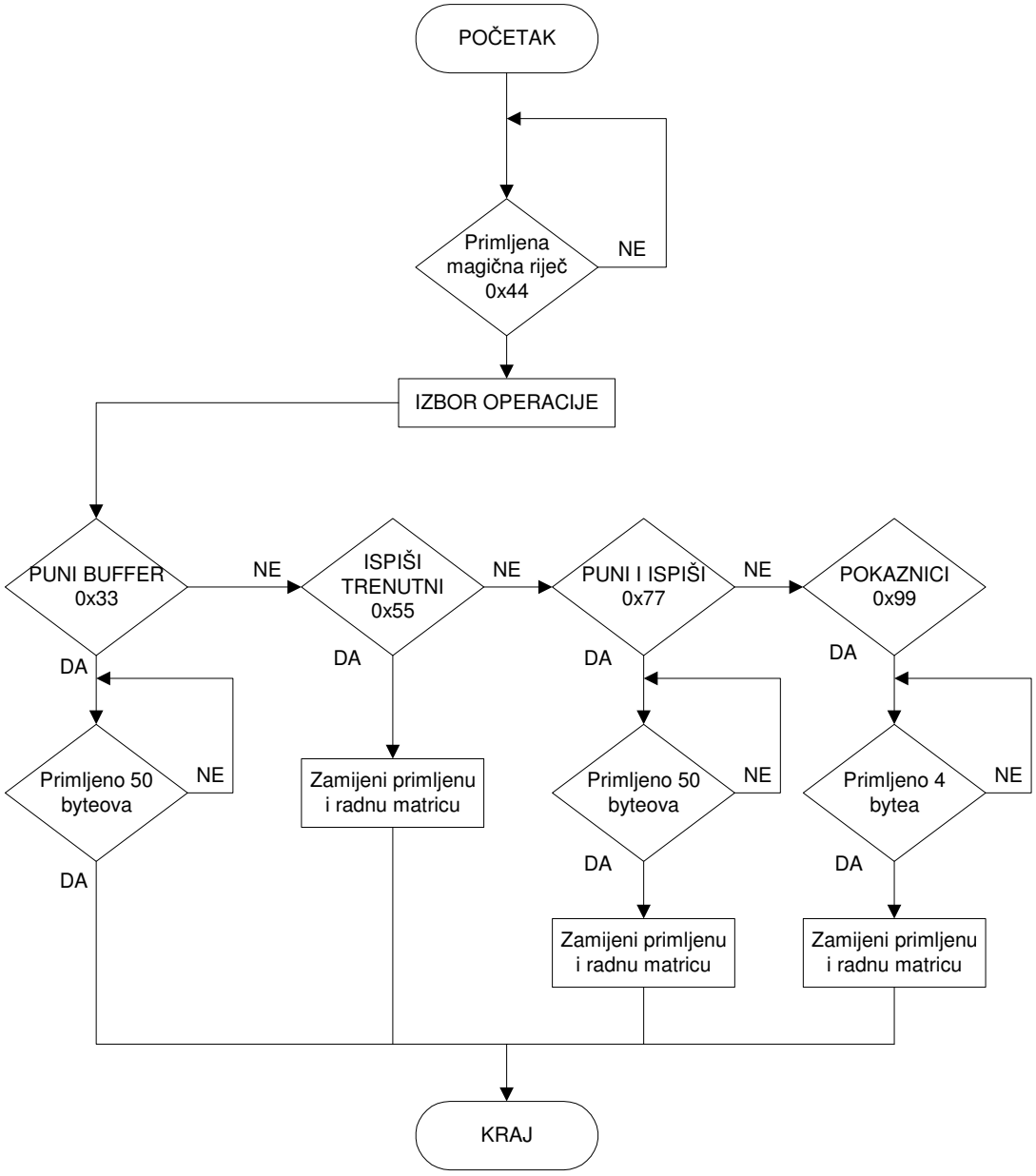
- programiranje AVR mikrokontrolera odvija se putem SPI protokola. To omogućuje sustavu ugradnju samo jednog konektora (X1) preko kojeg ide komunikacija s master sustavom i samo programiranje memorije mikrokontrolera (ukoliko je potrebno)
- osjetno veću brzina rada

Više o SPI sabirnici može se naći u [9]. Bitno je na *master* sustavu podesit iste SPI postavke kako bi se uspostavila veza. Za detalje proučiti *clock & phaze* odabir u programskom kôdu priloženom na CD-u uz ovaj završni rad.

Dijagram toka komunikacije prikazan je na slici 47. U nastavku slijedi kratak opis komunikacije.

1. Display je u stanju čekanja i ispisuje početnu sliku upisanu u njegovu ROM memoriju (sve diode upaljene što je ujedno i testiranje ispravnosti). Na 7-segmentnim pokaznicima piše broj 0x0000.
   Master sustav šalje **magičnu riječ 0x44** nakon koje display očekuje naredbu. Sve ostale podatke koje nisu magična riječ display zanemaruje.
2. Drugi primljeni *byte* određuje instrukciju.
   - **PUNI BUFFER (0x33)**
     Display sljedećih 50 pristiglih podataka (1 podatak je 1 byte) sprema u rezervnu matricu. Kako primi svih 50 podataka vraća se u stanje čekanja.
   - **ISPISI TRENUTNI (0x55)**
     Zamjenjuju se pokazivači na radnu (ona koja se trenutno ispisuje) i zadnju primljenu matricu. Ovo omogućuje master sustavu odgodu prikaza već poslane slike.
   - **PUNI I ISPISI (0x77)**
     Naredba koja objedinjuje gornje dvije u jednu. Koristi se kada master sustav konstantno osvježava sliku.
   - **POKAZNICI (0x99)**
     Display sljedećih 4 podatka sprema u rezervnu matricu (ova matrica nije ista kao za ispis slike na matricu dioda). Nakon primljenih 4 podatka display odmah postavlja nove vrijednosti na 7-segmentne pokaznike i prelazi u stanje čekanja.
3. Komunikacije je završila

Program je napisan u C-u. Korišteni **IDE** je *Atmelov AVR Studio*. C je preveden pomoću AVR-GCC prevodioca.



Slika 48. Dijagram toka komunikacije

Zaključak

Digitalna obrada signala sve više uzima maha. U ne tako davnoj prošlosti, većinu operacija nad signalom izvršavali su analogni sklopovi. Danas se, odmah nakon prijamnika, signal vodi na AD pretvornik iza kojeg se krije moćan DSP procesor. DSP procesor obavlja sve zadaće koje je u prošlosti obavljalo analogno sklopovlje. Analogni sklopovi su skupi i osjetljivi naspram digitalnih računala. Ipak, potrebno je napomenuti da analogni krugovi nikada neće potpuno izumrijeti u segmentu obrade signala. Uzorkovanje i rekonstrukcija signala nepredvidiv je postupak bez poznavanja *Shannonove* frekvencije signala, koju određuje analogni anti-aliasing filtar smješten između prijamnika i AD pretvornika.

U ovom završnom radu pokazano je, da je za jednostavniju analizu spektra signala, moguće koristiti komercijalne, jeftine i svima dostupne mikrokontrolere. Pri tome je za obradu odabran audio signal, jer je prisutan svugdje oko nas.

Analizator spektra je odličan odabir za završni rad studenata Fakulteta elektrotehnike i računarstva, jer potrebuje znanja iz mnogih područja koje smo slušali u protekle tri godine. Glavna područja koje kombinira su u prvom redu matematika, programiranje i elektronika.

Izrada projekta *displaya* temeljenog na matrici LED dioda, dokazuje da je moguće uz malo truda stvoriti kvalitetan display široke primjene uz malu cijenu.

Analizator spektra na ugradbenom računalu

Sažetak

U radu je opisana izrada dva odvojena elektronička sustava koja zajedno tvore analizator spektra audio signala. Pri tome nije izostala matematička podloga koja stoji iza postupka analize. Puno je pažnje posvećeno Fourierovoj transformaciji. Objašnjen je postupak uzorkovanja analognog signala i princip rada korištenog analogno-digitalnog pretvornika. Opisane su osnovne značajke korištenog mikrokontrolera, kao i način rada periferije neophodne za ovaj projekt. Obrada signala popraćena je mnogim slikama i odsječcima programskoga kôda.

Za potrebe prikaza spektra, predstavljen je projekt izrade pokaznika temeljenog na matrici LED dioda veličine 20 × 20 (eng. *Dot Matrix Display*). Fokus je stavljen na objašnjenje rada hardvera prikazanog nekolicinom električnih shema. Opisan je način komunikacije i instrukcije koje *display* podržava.

Summary

This work describes the development of two separate electronics systems which act together forming an audio spectrum analyzer. During the process, mathematical background wasn't neglected. Fourier transform represents the very core of the mathematical system. The work analyses analog signal sampling and describes the principles of analog to digital converters. Microcontroller analysis focuses on the operation basics, as well as the peripherals of the system. Signal processing is further described through pictures and source codes. A dot matrix display was introduced during the project for purpose of spectrum display. The focus of the work was put on describing the hardware which is shown with a couple of electric schemes. Display's communication and instructions were also described.

Literatura

[1] Wikipedia: Joseph Fourier
http://en.wikipedia.org/wiki/Joseph_Fourier

[2] Wikipedia: Dirichlet conditions
http://en.wikipedia.org/wiki/Dirichlet_conditions

[3] Nicholas Harrison: Fourier Series & Fourier Transforms, 2003

[4] R.N. Mutagi: Understanding the Discrete Fourier Transform
http://rfdesign.com/mag/401rfdf3.pdf

[5] Steven W. Smith: The Scientist and Engineer's Guide to Digital Signal Processing
http://www.dspguide.com/ch12/2.htm

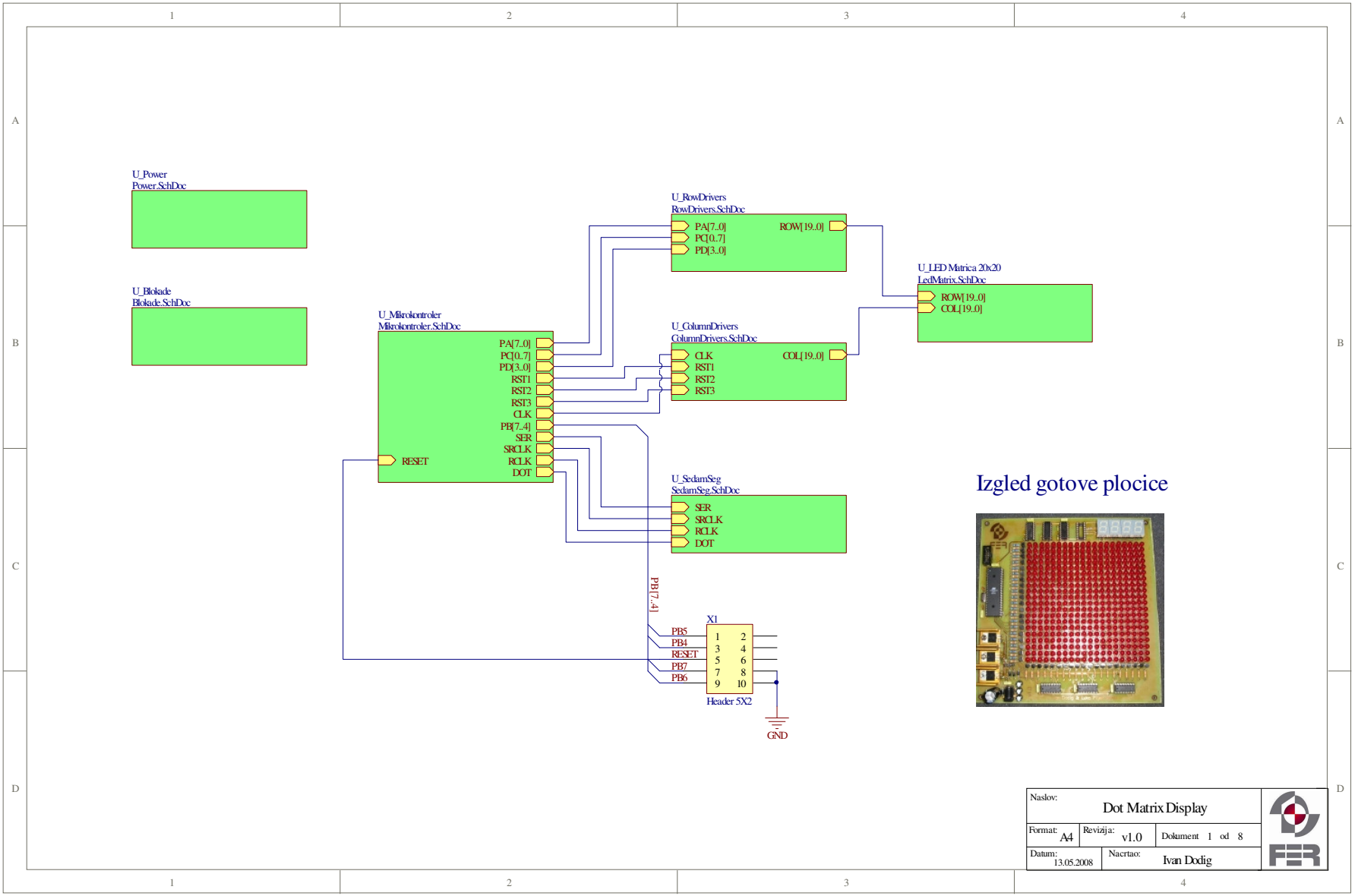
[6] Atmel: AT91SAM7X256 *Product information & datasheets*
http://www.atmel.com/dyn/products/product_card.asp?part_id=3755

[7] Tomislav Gracin: DIPLOMSKI RAD br. 1074 / lipanj 2007.
«*Sklopovska i programska jezgra inteligentnog mjernog sustava opće namjene*»
- za diplomski rad kontaktirati ZESOI@FER

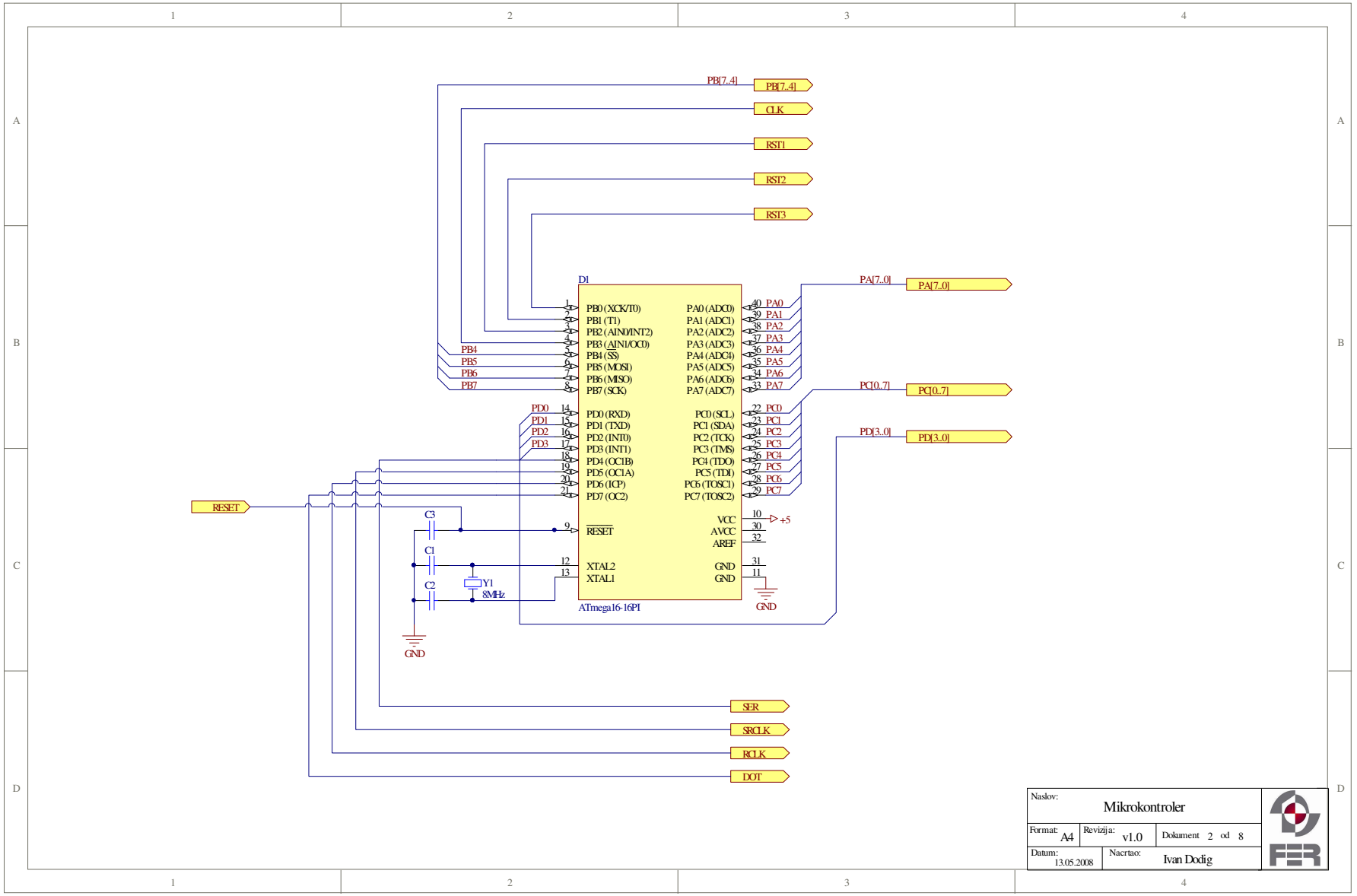
[8] Wikipedia: Window function
http://en.wikipedia.org/wiki/Window_function

[9] *Atmel Corporation*: ATmega16 *datasheet*
http://www.atmel.com/dyn/products/product_card.asp?part_id=2010

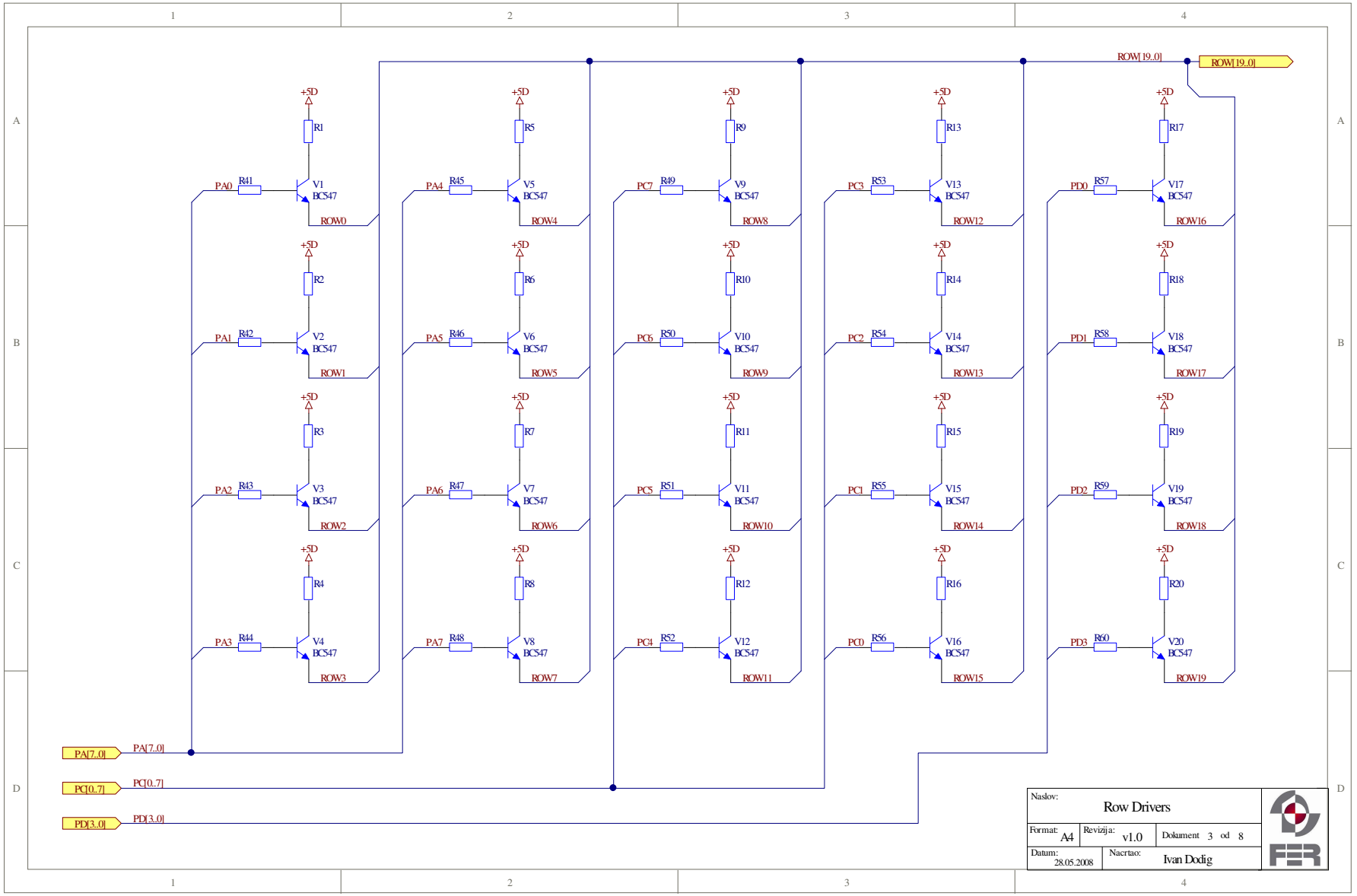
Dodatak A – Električne sheme za *Dot Matrix Display*



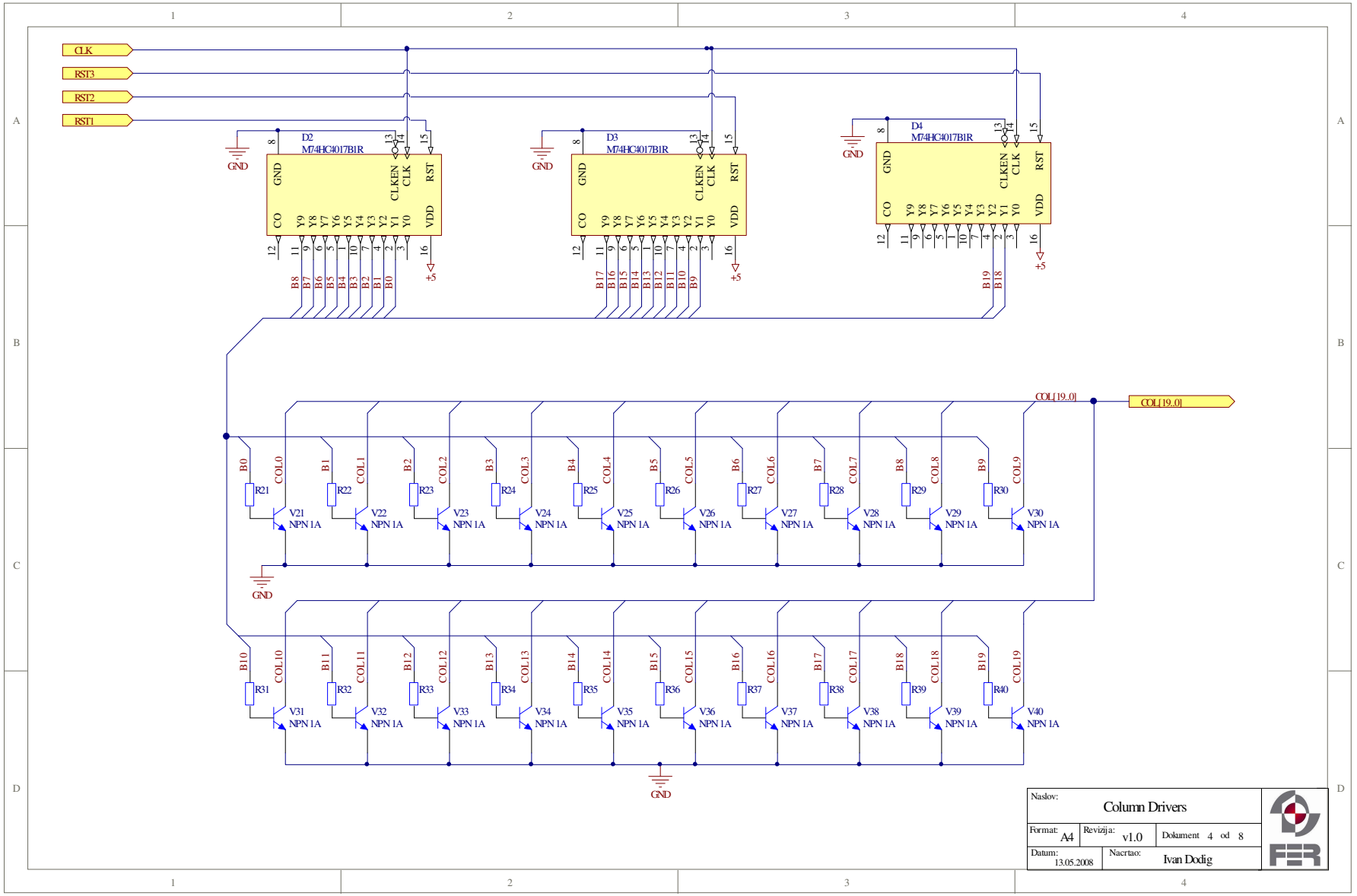
Dodatak A – Električne sheme za *Dot Matrix Display*



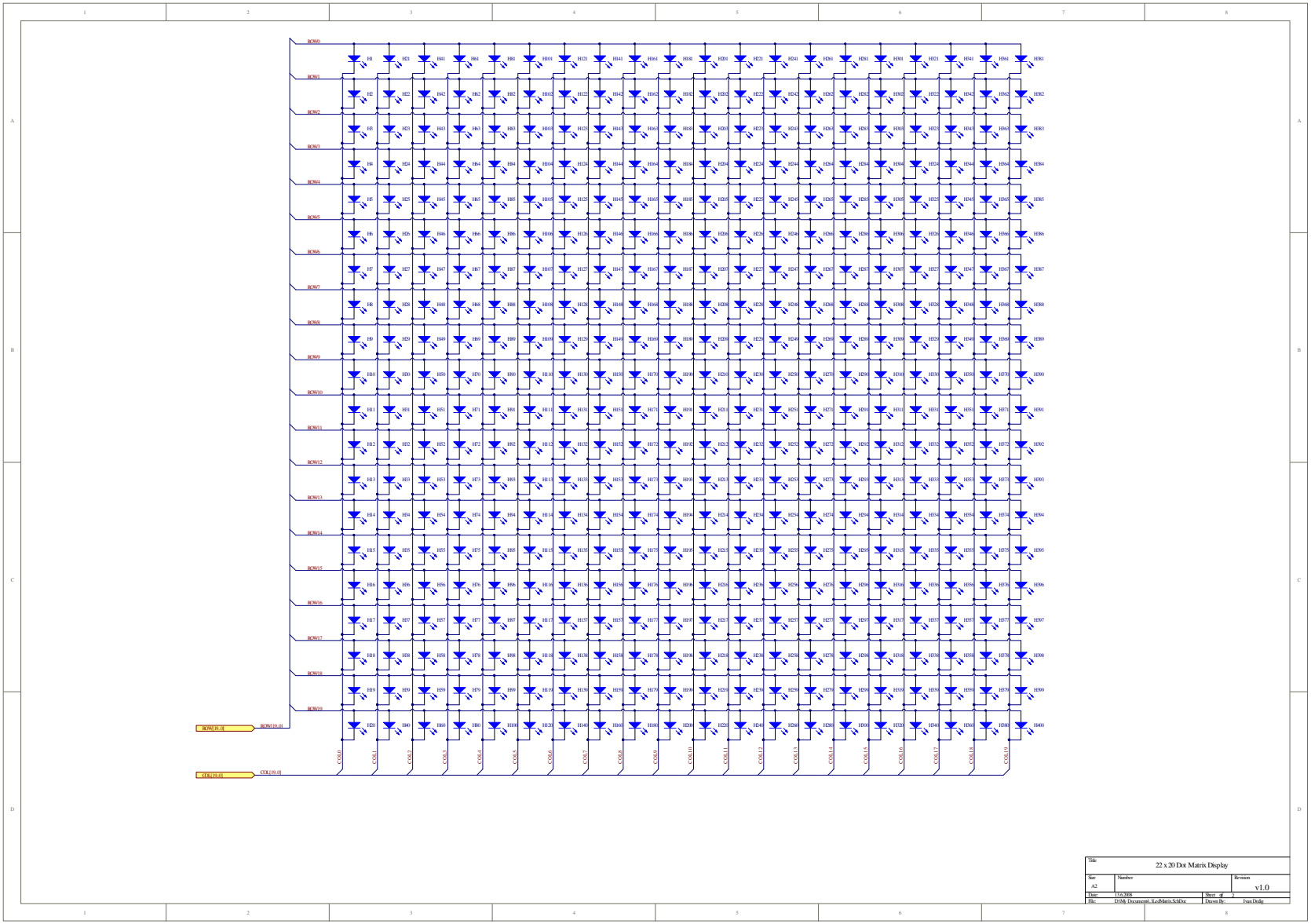
Dodatak A – Električne sheme za *Dot Matrix Display*



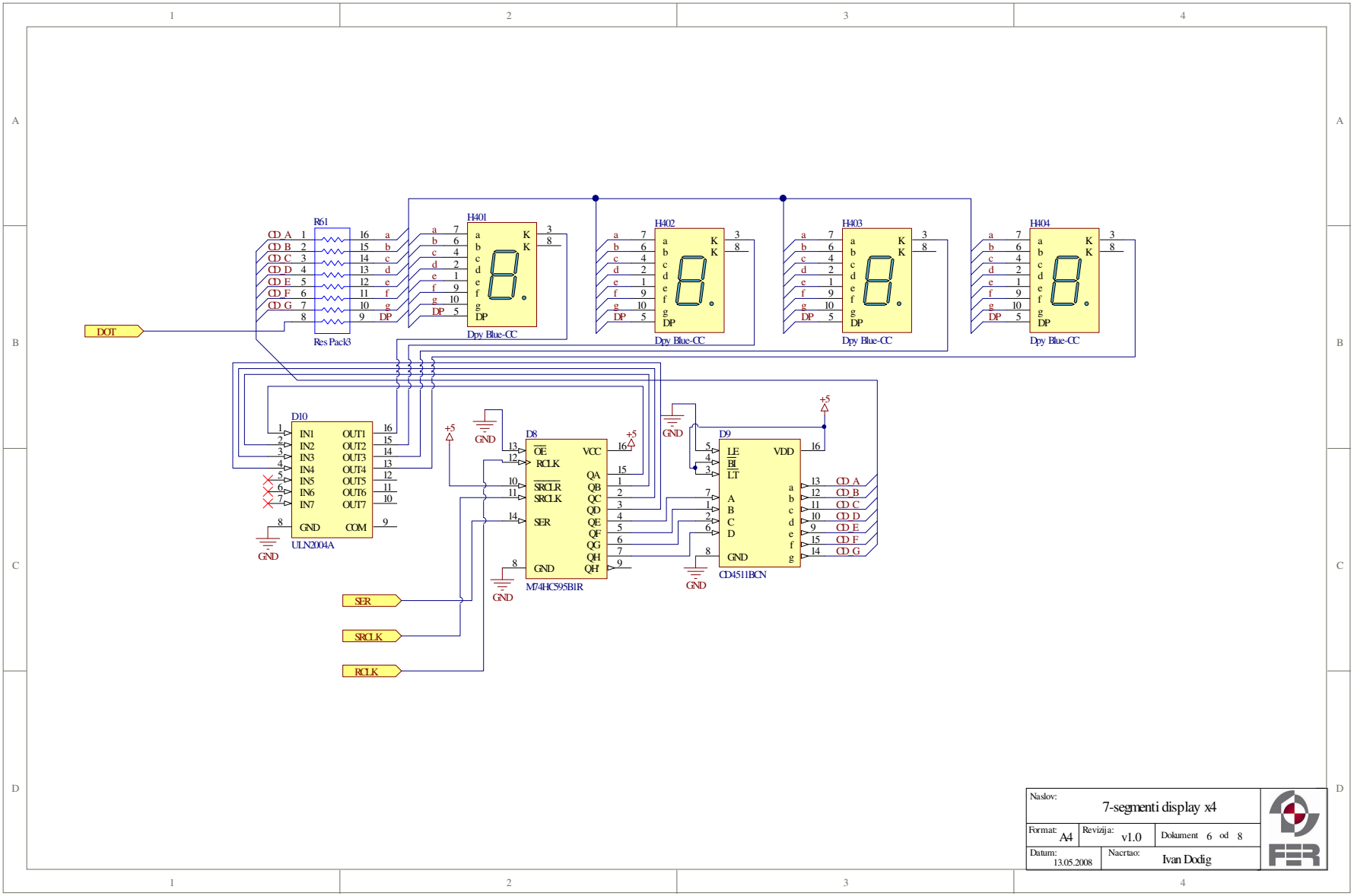
Dodatak A – Električne sheme za *Dot Matrix Display*



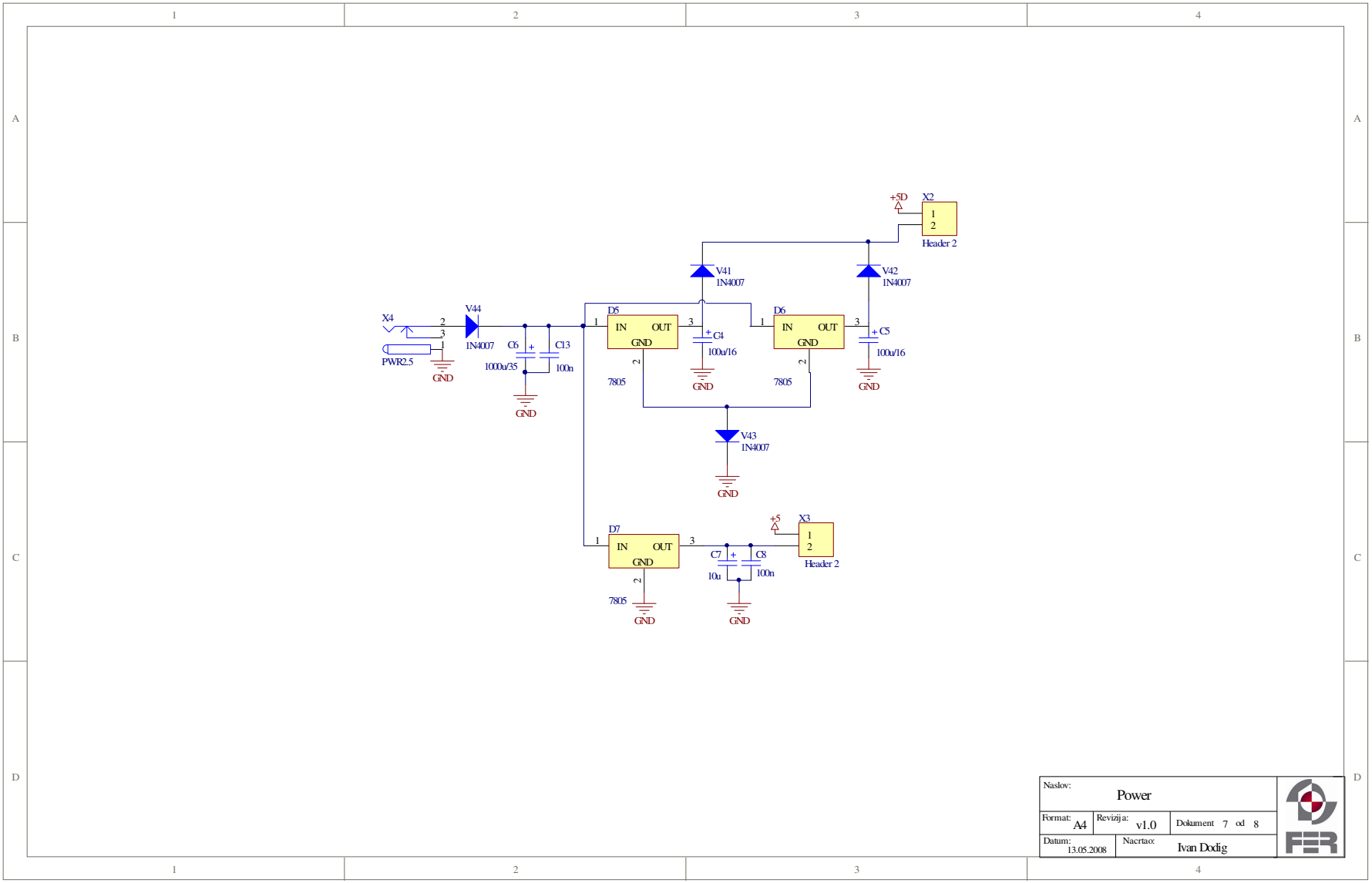
Dodatak A – Električne sheme za *Dot Matrix Display*



Dodatak A – Električne sheme za *Dot Matrix Display*



Dodatak A – Električne sheme za *Dot Matrix Display*



Dodatak A – Električne sheme za *Dot Matrix Display*