Sveučilište u Zagrebu
Fakultet strojarstva i brodogradnje

DIPLOMSKI RAD

Mentor:
Prof. dr. sc. Mario Essert

Antonio Magdić

Zagreb, 2007.

Svojoj voljenoj supruzi Nini
posvećujem

Sažetak

Jedan od osnovnih tehničkih zahtjeva moderne tehničke dokumentacije je standardizacija tehničkih shema. Danas poznajemo raznovrsna rješenja pridružena različitim CAD aplikacijama. Međutim, on-line pristup generiranju tehničkih shema posve je zanemaren. U ovom radu želi se načiniti rješenje koje će zadovoljiti nastavne potrebe iz temeljnih kolegija koji se slušaju na usmjerenju Mehatronika i robotika Fakulteta strojarstva i brodogradnje Sveučilišta u Zagrebu, s mogučnošću njegova proširenja na nove sadržaje.

Ključne riječi: tehničke sheme, M4, PIC, LaTeX, Scriptrunner, PGF, Tikz

Sadržaj

Popis slika

Popis tablica

Popis oznaka

Oznaka		Značenje oznake
C	[F]	Kapacitet
f_{poc}	[Hz]	Početna frekvencija
f_{kr}	[Hz]	Krajnja frekvencija
L	[H]	Induktivitet
K_{D}		Derivacijsko pojačanje
K_{I}		Integralno pojačanje
K_{P}		Proporcijalno pojačanje
M	[kg]	Masa
R	[Ω]	Električni otpor

Kratice

CAD	Computer Aided Design
DVI	DeVice Independent
EPS	Encapsulated Postscript
GIF	Graphics Interchange Forma
GNU	GNU's Not Unix
MPS	MetaPost
PDF	Portable Document Format
PGF	Portable Graphic Format
PHP	Hypertext Preprocessor
PNG	Portable Network Graphics
PS	Postscript
TikZ	TikZ ist kein Zeichenprogramm
TROFF	The Text Processor for Typesetters

Zahvala

Zahvaljujem se mojim roditeljima na maksimalno mogućoj potpori, pomoći i razumijevanju za sve moje situacije tijekom studiranja.

Zahvaljujem se svom mentoru, Prof. dr. sc. Mariu Essert, što mi je omogućio podršku i što mi je omogućio izradu ovog diplomskog rada, zahvaljujem mu na kvalitetnoj stručnoj pomoći, korisnim komentarima i savjetima.

Na kraju, posebna zahvala mojoj voljenoj supruzi Nini bez čije podrške i ljubavi ne bi bilo ovoga rada.

Izjava

Izjavljujem da sam diplomski rad na temu "*Generiranje interaktivnih tehničkih shema*" izradio samostalno koristeći navedenu literaturu i znanje stečeno tijekom studija. Stručnu pomoć u odabiru literature, te korisne savjete tijekom izrade svesrdno mi je pružio mentor Prof. dr. sc. Mario Essert.

1 Uvod

Danas je generiranje tehničkih shema postalo sve lakši posao za inženjere. Koristeći neki od programskih paketa kao što su AutoCAD, Catia, SolidWorks, Pro-Engineer, Microsoft Visio i mnogi drugi postavlja se pitanje treba li nam uopće neko nekomercijalno on-line rješenje?

Zamislimo da trebamo nacrtati jednu jednostavnu tehničku shemu iz područja elektrotehnike ili automatske regulacije. Da bi to bilo moguće na računalu trebamo imati instaliran neki programski paket koji bi nam to omogućio, npr. AutoCAD ili Microsoft Visio. Ti programski paketi pružaju mnogo opcija pa krajnjem korisniku (početniku) mogu stvarati problem u crtanju jednostavne sheme, bez poznavanja osnovnih naredbi. Korisnik je obasut viškom informacija (ikona) među kojima se treba snaći i pronaći njemu bitne, što može biti mukotrpan posao. Tek nakon utrošenog vremena u proučavanju same aplikacije (pa čak i višednevnog tečaja) može započeti crtanje sheme. Time se gubi puno vremena, dok crtanje same sheme iako jednostavne može iziskivati mnogo napora.

Želi se stoga načiniti on-line aplikacija koja bi omogućila uštedu vremena i jednostavnost pristupa. Želi se na jednostavan način, sa samo nekoliko programskih naredbi i bez potrebe proučavanja dodatne literature postići isti cilj. Također nije potrebno posjedovati jedan od skupih programskih paketa. Dovoljan je samo web preglednik (*Mozilla Firefox*, *Internet Explorer*) i internet veza, što je danas prisutno na gotovo svakom računalu.

Aplikacija će biti modul za već postojeći sustav *Scriptrunner*[1] koji godinama razvijamo, u svrhu poboljšanja nastave na našem i drugim fakultetima i ustano-

[1] http://scriptrunner.fsb.hr/

vama.

Također će se za potrebe elektrotehnike, automatske regulacije, hidrauličkih pogona i računalnih praktikuma definirati skupine elemenata s kojima se sheme generiraju. To će uvelike olakšati crtanje shema iz tih područja.

Sheme će se generirati korištenjem M4[2] makroprocesora, te DPIC[3] procesora koji je u stanju pretvoriti dobivene sheme u neki grafički format (npr. LaTeX, PGF, PDF, GIF, PNG). Tako dobivena datoteka nakon preuzimanja može se uključiti u bilo koji dokument.

U svakom poglavlju obradit će se nekoliko primjera koji će olakšati razumijevanje i korištenje ove on-line aplikacije.

U prilogu ću opisati instalaciju svih potrebnih programskih alata za Linux platfomu, kao i leksičku strukturu PIC jezika. Popis korištenih oznaka i kratica dat je na početku, a popis korištene literature na kraju ovog rada.

[2] http://www.gnu.org/software/m4/

[3] http://ece.uwaterloo.ca/ aplevich/dpic/

2 Crtanje slika pomoću GNU PIC programa

PIC je programski jezik za definiranje dijagrama preko objekata kao što su okviri (eng. *boxes*) međusobno povezanih linijama ili strelicama. Jezik je zamislio i napisao Brian Kernighan, tvorac C jezika, 1991. u Bell laboratorijima. Njegov PIC compiler prevodi i izvodi ovakav opis dijagrama u konkretne naredbe za crtanje. PIC je proceduralni programski jezik, koji uključuje pridruživanje vrijednosti varijablama, ima uvjetne naredbe i petlje, te grupiranje naredbi u macro naredbe. PIC je prvi put načinjen kao preprocesor TROFF[1] sustava za obradu dokumenata, a kasnije je u GNU[2] inačici proširen na TEX, odnosno LATEX dokumente. Sam compiler doživio je i neka proširenja i prilagodbe (gpic, dpic).

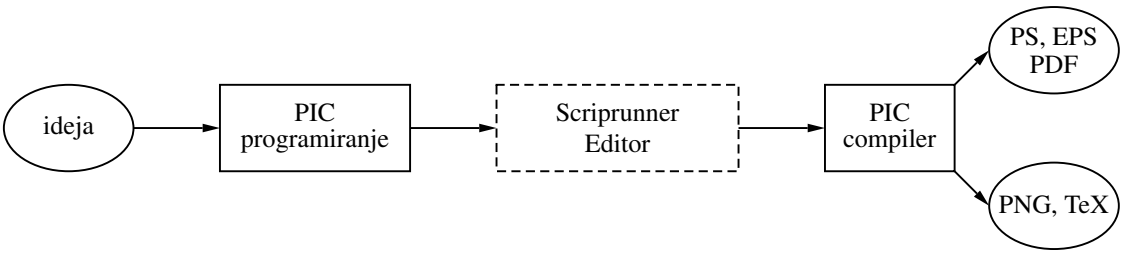
Tako na primjer, DPIC - Dwight Aplevich-eva inačica PIC-a generira slike samostalno, ali može raditi i kao preprocesor za predobradbu unutar sustava za tvorbu dokumenata.

[1] The Text Processor for Typesetters
[2] GNU's Not Unix

2.1. Osnove PIC programa

Slike u PIC programu opisuju se proceduralno, kao skupovi objekata povezanih u nizu. Ako se drugačije ne odredi, PIC pokušava povezati objekte u nizu s lijeva nadesno, povezujući ih vizualno prirodnim slijedom. Slijedi program što će kao rezultat dati dijagram prikazan na slici 2.1.

```
.PS
ellipse "ideja"
arrow
box width 1.1 "PIC" "programiranje"
arrow
box width 1.4 "Scriprunner" "Editor" dashed
arrow
A: box "PIC" "compiler"
arrow up 0.2 right 0.2 from A.ne
ellipse "PS, EPS" "PDF"
arrow down 0.2 right 0.2 from A.se
ellipse "PNG, TeX"
.PE
```



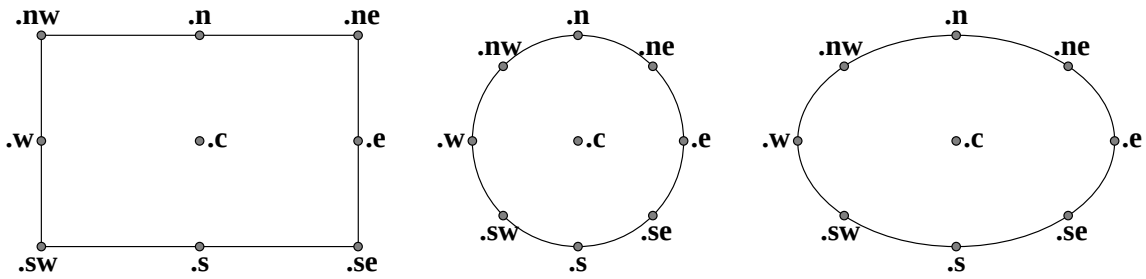
Slika 2.1: PIC programiranje

Iz programa je vidljivo da svaki PIC program za crtanje slika započinje s **.PS** (*program start*) i završava s **.PE** (*program end*) naredbom. Unutar ovih naredbi pišu se naredbe za opisivanje, pozicioniranje i crtanje objekata. U gornjoj slici objekti su elipsa (engl. *ellipse*), okvir ili pravokutnik (engl. *box*) i strijelica (engl. *arrow*). Razlikujemo zatvorene (elipsa/krug, pravokutnik/kvadrat) i otvorene objekte (linija, strijelica, luk i spline).

Objekti se opisuju atributima, pa će npr. atribut **dashed** okvir nacrtati crtkano, a atribut **width** u jednom će slučaju proširiti okvir za 40% (1.4), a u drugom za 10% (1.1) kako bi tekst u njemu bio prikladno upisan. Tekst ili string (niz znakova unutar dvostrukih navodnika) u objekte se upisuje u redovima, od vrha prema dnu, a u naredbi se odjeljuje prazninom.

Varijabla (u gornjem slučaju **A:**) služi za označavanje objekta, koji se kasnije može pozvati preko njenog imena. Ime varijable je alfanumerički string koji počinje velikim slovom, a završava dvotočkom.

Pomak nevidljivog kazala na čijem mjestu se crta objekt pomiče se po crtaćoj plohi naredbom **move** (za $\frac{1}{2}$ incha) ili nekom naredbom za crtanje određenog objekta. Smjer crtanja mijenja se relativno s obzirom pozicije kazala naredbama: **up** - gore, **down** - dolje, **left** - lijevo, **right** - desno ili s obzirom na navigacijske točke, koje ima svaki zatvoreni objekt (slika 2.2).

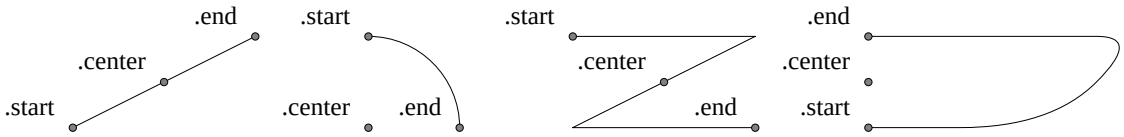


Slika 2.2: Navigacijske točke zatvorenog objekta

Kratice se odnose na strane svijeta (**.n** za north - sjever, **.s** za south - jug, **.e** za east - istok i **.w** za west - zapad) i njihove međuodnose. Ove navigacijske oznake mogu se zamijeniti relacijskim oznakama: **.top**, **.bottom**, **.left** i **.right**, koje predstavljaju sinonime za **.n**, **.s**, **.e** i **.w** respektivno, te se mogu čak i kratiti kao **.t**, **.b**, **.l** i **.r**.

Želimo li crtati objekte odozgo prema dolje, niz počinje naredbom **down**; odozdo prema gore, naredbom **up**, a s desna ulijevo naredbom **left**. Naredba **right** je pretpostavljena i nije ju nužno navoditi kao početnu.

Kod otvorenih objekata razlikuje se početak (**.start**), sredina (**.center** ili **.c**) i kraj (**.end**).



Slika 2.3: Navigacijske točke otvorenog objekta

Atributi **at** - na, **from** - od, **to** - prema i **with** - sa koriste se u naredbama za rad s pozicijama objekta. Promjena smjera crtanja zakrivljenih, otvorenih objekata postiže se naredbom **cw** - za promjenu u smjeru kazaljke na satu ili **ccw** - za promjenu u smjeru suprotnom smjeru kazaljke na satu.
Tako će sljedeći program:

```
.PS
line;
arc;
arc cw;
line;
move "kraj"
.PE
```

nacrtati primjer prikazan slikom 2.3.

Slika 2.4: Primjer otvorenog objekta

Naredba s tekstom na kraju pomaka ili nacrtanog objekta ispisuje taj tekst na mjestu nevidljivog kazala.

Dakako, PIC koristi Kartezijev koordinatni sustav s koordinatom (0,0) u lijevom kutu zamišljene ravnine na kojoj se objekti crtaju. Apsolutne koordinate opisuje se dakle (X,Y) parom. X koordinata se pritom povećava udesno, a Y prema gore. Međutim, apsolutne koordinate u PIC-u treba izbjegavati, sve je lakše (pogotovo promjene) izvoditi u opisanim relativnim koordinatama.

2.2. Veličine objekata

Postoje pretpostavljene veličine PIC objekata. Tablica 2.1 prikazuje vrijednosti tih pretpostavljanih (engl. *default*) veličina izraženih u inch-ima (").

Tablica 2.1: Pretpostavljene veličine PIC objekata

Objekt	**Pretpostavljena vrijednost**
box	0,75" širina x 0,5" visina
circle	0,5" promjer
ellipse	0,75" širina x 0,5" visina
arc	0,5" radius
line	0,5" širina
arrow	0,5" širina

Globalnom varijablom **scale** mogu se mijenjati odnosi veličina, pa će *scale=2.54* promijeniti skalu u centimetre, odnosno *scale=25.4* u milimetre.

Objekti okviri i elipse mijenjaju svoje veličine promjenom atributa **width** (kratica **wid**) za širinu, odnosno atributom **height** (kratica **ht**) za visinu.

Objekti krug i luk mijenjaju svoju veličinu promjenom polumjera **radius** (kratica **rad**). Krug može mijenjati i svoj promjer atributom **diameter** (kratica **diam**).

Slika 2.5: Promjena veličine kruga i luka naredbom **rad**

I tekst se može zamišljati kao da je okružen nevidljivim pravokutnikom, čija se širina može mijenjati atributom **textwid**, a širina **textht**, jer nema drugačije eksplicitne promjene veličine fonta alfanumeričkih znakova.

Upis stringa u okvir, ne povećava automatski veličinu okvira, nego se to mora postići naredbama, npr. *box width 3* povećat će širinu okvira tri puta, pa će ako je pretpostavljena vrijednost bila 1 cm, nova vrijednost biti 3 cm.

Moguće je također mijenjati debljinu linije objekta. To se postiže upravljanjem globalne varijable **linethick** na razini sustava (u TeX modu ona je jednaka 8 miliinch-a, a u TROFF sustavu proporcionalna je veličini točke). Najlakši način mijenjanja debljine linije je preko atributa **thickness** (kratica **thick**). Na primjer, *circle thick 1.5* nacrtat će krug pretpostavljenog polumjera s debljinom kružnice od 1.5 točke. Na debljinu linije objekta ne utječe **scale** varijabla.

Naredba **same** (isti) služi za specifikaciju iste veličine novog objekta kakav je bio prethodni istog tipa.

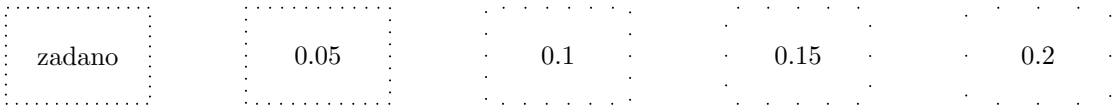
2.3. Dekoriranje objekata

Objekti se dekoriraju atributnim naredbama, kao što su:

- **dashed** - za crtkane objekte
- **dotted** - za točkaste objekte
- **color** - za boju objekta
- **shaded** - za boju unutarnje površine zatvorenog objekta
- **outline** - za boju vanjskog ruba zatvorenog objekta
- **filled** (kratica **fill**) - za ispunjavanje (nijansama sive boje) površine zatvorenog objekta

GNU gpic dozvoljava da se elipse, kružnice, lukovi i okviri prikazuju crtkanim ili točkastim a ne punim linijama. To se može postići korištenjem atributa **dotted** za crtkane ili atributa **dotted** za točkaste linije. Za mijenjanje razmaka između crtica ili točaka koristimo broj nakon naredbe. Slika 2.6 prikazuje nekoliko vrsta crtkanih dok slika 2.7 točkastih objekata.

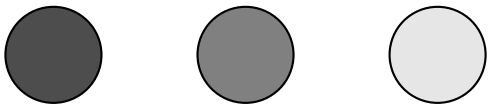
Slika 2.6: Primjer crtkanih objekta



Slika 2.7: Primjer točkastih objekta

Također je moguće objekte ispuniti nijansama sive boje. Slika 2.8 kreirana je korištenjem sljedećeg programskog koda:

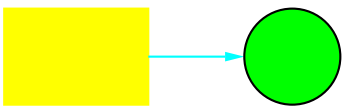
```
.PS
circle fill; move;
circle fill 0.5; move;
circle fill 0.9;
.PE
```



Slika 2.8: Primjer ispunjenih objekta

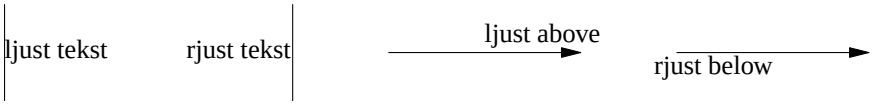
Želimo li pak objekte crtati određenom bojom, to možemo učiniti atributom **color**. Slika 2.9 prikazuje nekoliko objekata u različitim bojama, a kreirana je sljedećim programskim kodom:

```
.PS
box color "yellow";
arrow color "cyan";
circle shaded "green" outline "black";
.PE
```



Slika 2.9: Primjer bojanih objekta

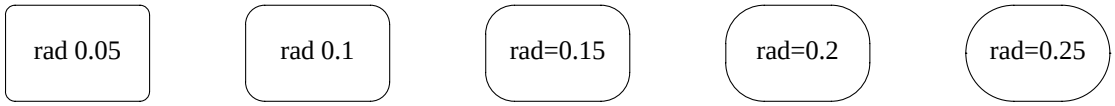
Tekst se centrira unutar pridruženog geometrijskog objekta. Atributom **ljust** moguće je pozicionirati tekst na odgovarajuću točku iza koje će biti napisan udesno. Na sličan način atribut **rjust** pozicionira tekst na desnu koordinatu (a sam tekst se ispisuje ulijevo). Atributi **above** i **below** centiraju tekst na polovicu udaljenosti od linije u zadanom smjeru.



Slika 2.10: Centriranje teksta unutar objekta

U GNU gpic-u objekti mogu imati **aligned** atribut. Neki tekst u objektu s tim atributom bit će rotiran oko središta objekta u smjeru od početne do krajnje točke objekta.

Također u GNU *gpic*-u moguće je mijenjati okvir zaobljivanjem krajeva okvira kako je prikazano na slici 2.11.



Slika 2.11: Povećanje vrijednosti polumjera naredbom **box rad**

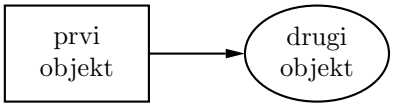
Linije i lukovi dekoriraju se strijelicama. Naredba **arrow** nije ništa drugo nego naredba **line ->**. Liniju je dakako moguće označiti i na oba kraja s naredbom **line <->**. Strijelice imaju atribute **width** i **height** s kojima se upravlja njihovom dužinom i nagibom. Tip strijelica unpravlja se stilskom varijablom **arrowhead**.

2.4. Imenovanje objekata

Imenovanje objekata provodi se već spomenutim oznakama (labelama objekta) na koje se druge naredbe mogu pozivati specifikacijom posebnih mjesta (navigacijskih točaka) tog objekta.
Tako će program:

```
.PS
A: box "prvi" "objekt"
move;
B: ellipse "drugi" "objekt"
move;
arrow right at A .r;
.PE
```

deklarirati objekte s oznakama **A** i **B** koje će povezati strijelicom (**arrow**) s objekta **A** (**at A**) udesno (**right**) na zapad (**.w**) odnosno **.r** (right) objekta **B**:



Slika 2.12: Imenovanje objekata

Labele ili oznake nisu konstante, nego varijable, pa se njihova pozicija može mijenjati, npr *A: A + (1,0)* pomaknut će poziciju za 1 jedinicu udesno (po x varijabli, jer je u ovom slučaju y=0).

Osim imenom, objekti se mogu dohvaćati i njihovim redoslijedom crtanja, pa će *3rd ellipse* označavati treću nacrtanju elipsu, **2nd last box** značiti predznadnje nacrtani okvir (drugu od zadnje), dok će **5th** odgovarati petom po redu napisanom stringu. Općenito, *n***th** označuje n-ti po redu objekt. Moguće je pritom specificirati i točniju poziciju objekta, pa će **last circle .s** značiti donju točku, dno (ili jug) posljedne nacrtane kružnice.

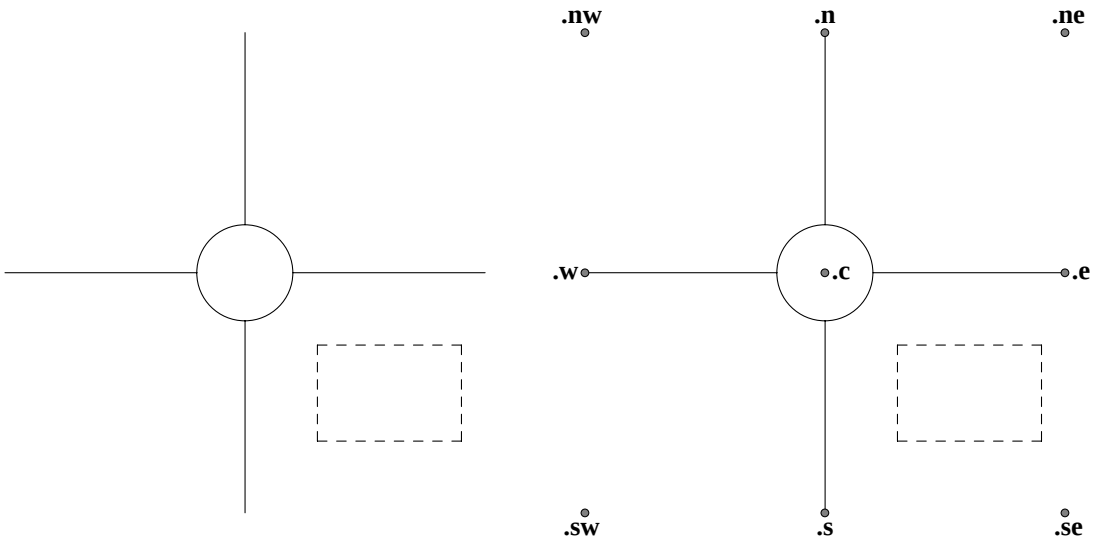
Više objekata može se povezati u *kompozitni objekt*. U tom slučaju objekti elementi pišu se unutar uglatih zagrada, a kompozitni objekt se referencira kao []. Referenca na 4-ti kompozitni objekt bit će tako **4th**[].

Kompozitni objekt stvara se nizom naredbi obuhvaćenih uglatim zagradama i označenim nekom oznakom (labelom). Može se promatrati kao jedinstveni zatvoreni objekt s vlastito definiranim oblikom i veličinom.

Program:

```
A: [
    circle;
    line up 1 at last circle .n;
    line down 1 at last circle .s;
    line right 1 at last circle .e;
    line left 1 at last circle .w;
    box dashed with .nw at last circle .se + (0.2, -0.2);
    Caption: center of last box;
]
```

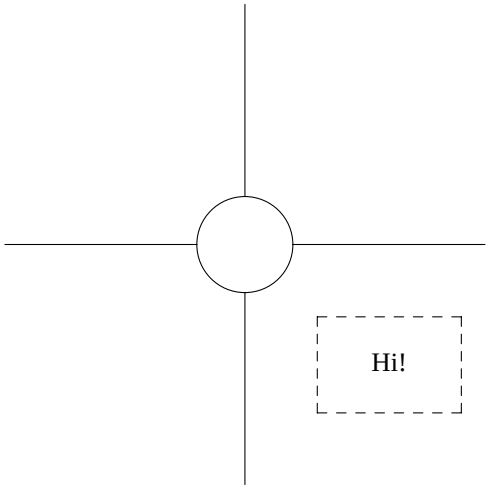
daje blok prikazan slikom 2.13.



Slika 2.13: Kompozitni objekt

Referenciranje na kompozitni objekt isto je kao na bilo koji drugi objekt, na primjer **A.s** dohvaća jug (**.s**) objekta, ali također i sa **last[].s** ako je ovaj objekt zadnji nacrtan. Kako se vidi, ovakvi objekti odgovaraju klasama u objektno orjentiranom jeziku.

Na sve varijable definirane unutar objekta moguće se je referencirati izvana, pa će **A.Caption** ili **last[].Caption** odnositi se na središte okvira definiranog u bloku **A**. Naredbom **"Hi!" at A.Caption** dobit će se objekt prikazan slikom 2.14.



Slika 2.14: Referenciranje na kompozitni objekt

Blokovi se mogu gnijezditi, čime se stvara složena struktura dijagrama:

```
.PS
P: [box "foo"; ellipse "bar"];
Q: [
     [box "baz"; ellipse "quxx"]
     "random text";
   ]
arrow from 2nd last [];
.PE
```

Gnijezdo ide iznutra prema vani, pa će strijelica u zadnjoj naredbi biti pridružena objektu **P**, a ne objektu **Q**.

2.5. Postavljanje (pozicioniranje) objekata

Postavljanje objekta na neku (x,y) koordinatu moguće je ostvariti apsolutnim i relativnim operacijama ili njihovim kombinacijama. Tako će naredba **last box .ne + (0.1, 0)** odrediti poziciju nevidljivog kazala uvećanu za **0.1** osnovne jedinice u x-smjeru (y se neće mijenjati, jer je argument jednak **0**) i to od sjevero-istočne pozicije zadnje nacrtanog bloka (okvira). Pozicija se može mijenjati i oduzimanjem apsolutnih iznosa na zadanoj poziciji.

Postoji specijalna oznaka **Here** (ovdje) koja se odnosi na trenutačnu poziciju kazala.

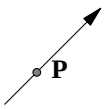
U izrazu se može koristiti visina, širina, polumjer, te **x** i **y** koordinate bilo kojeg objekta.
Na primjer:

```
A.x                 # x koordinata središta od A
A.ne.y              # y koordinata sjeveroistočnog kuta od A
A.wid               # širina od A
A.ht                # visina od A
2nd last circle.rad # polumjer predzadnje nacrtane kružnice
```

Pozicija se može određivati i interpolacijom između dviju zadanih pozicija. Sintaksa je "*razlomak* **of the way between** *pozicija1* **and** *pozicija2*" ili skraćeno "*razlomak* <*pozicija1*, *pozicija2*>".
Tako će naredba:

```
1/3 of the way between last arrow .start and last arrow .end
```

pozicionirati točku P kako je prikazano na slici 2.15.

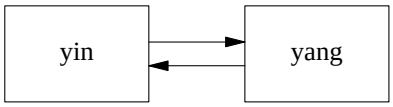


Slika 2.15: Primjer pozicioniranja točke na strelici

Dok će program:

```
.PS
A: box "yin"; move;
B: box "yang";
arrow right at 1/4 <A.e,A.ne>;
arrow left  at 1/4 <B.w,B.sw>;
.PE
```

kao rješenje dati objekt prikazan slikom 2.16.



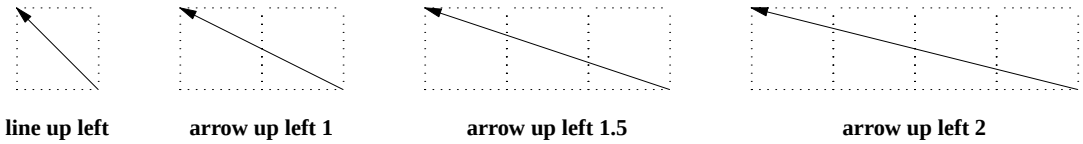
Slika 2.16: Primjer dvostruko povezivanih okvira

S pomoću dvije zadane pozicije **p** i **q** moguće je definirati poziciju **(p, q)** koja ima X koordinatu od **p** i Y koordinatu od **q**. Na isti način pozicija **(q, p)** imala bi X koordinatu od **q**, a Y koordinatu od **p**, kako pokazuje slika:

Slika 2.17: Korištenje (x, y) koordinata

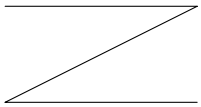
Ako se niz naredbi ili objekata obuhvati običnim () ili vitičastim {} zagradama, svi pomaci unutar ovakvog bloka se zaboravljaju i nakon zatvorene zagrade kazalo se nalazi na istoj poziciji na kojoj je bio prilikom ulaska u blok.

Zadavanje dijagonalnih linija ili strijelica postiže se upotrebom višestrukih **up**, **down**, **left** i **right** naredbi. Množitelj njihovog pomaka uvećava ili smanjuje zadani proj puta temeljnu duljinu okvira.
Primjer dijagonalnih linija pokazuje slika 2.18.



Slika 2.18: Primjer dijagonalnih linija

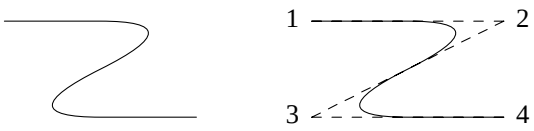
Ako se objekt sastoji od više linija ili strijelica, onda se za njegovo definiranje koristi naredba **then** (*onda*) kojom se odjeljuju pojedini segmenti.



Slika 2.19: line right 1 then down .5 left 1 then right 1

Zaobljenje linija postiže se naredbom **spline**. Slika 2.20 prikazuje primjer zaobljene linije nacrtane pomoću slijedećeg programa:

```
.PS
spline right 1 then down .5 left 1 then right 1
.PE
```

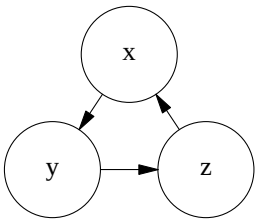


Slika 2.20: Primjer zaobljene linije

Kada se crtaju linije između kružnica tako da se ne presijecaju na navigacijskim točkama, onda se one skraćuju na polumjer kružnica na oba kraja. Za to služi naredba **chop**. Slijedeći primjer prikazan je slikom 2.21.

```
.PS
circle "x"
circle "y" at 1st circle - (0.4, 0.6)
circle "z" at 1st circle + (0.4, -0.6)
arrow from 1st circle to 2nd circle chop
arrow from 2nd circle to 3rd circle chop
arrow from 3rd circle to 1st circle chop
.PE
```

Chop atribut pomiče strijelice umjesto da se na njih nastavlja. On crta liniju iz središta kruga, ali je crta samo od radijusa kružnice prema drugom središtu, ali samo do oboda drugog kruga.



Slika 2.21: Primjer korištenja **chop** atributa

Moguće je utjecati na crtež upotrebom višestrukog chop operatora i definiranjem polumjera do kojeg se (ili od kojeg se) strijelica crta, npr. **line** ... **chop** *r1* **chop** *r2*.

Na koncu treba spomenuti da se objekt može načiniti nevidljivim, koristeći naredbu **invis** ili **invisible**. To je korisno u primjenama kada se između jednakih postojećih objekata ostavlja prostor za još jednog istih dimenzija.

2.6. Programiranje u PIC-u

Do sada su pokazane pojedinačne narede i njihovo grupiranje koje DPIC razumije i izvodi. Međutim, PIC je još složeniji programski jezik za crtanje grafike, jer uključuje pridruživanje, petlje i funkcije za stvaranje složenih grafičkih oblika.

Pridružba (engl. *assignment*) znači da se proizvoljnoj varijabli (koja počinje slovom, a sastoji se od niza alfanumeričkih znakova) pridružuje desna strana (od lijeve odijeljena znakom jednakosti) koja u sebi sadrži neki aritmetičko-logički izraz. Izraz koristi aritmetičke operatore: +, -, ∗, /, ^ i %, a kod gpic-a (GNU pic) mogu se još koristiti i logički operatori: !, &&, ||, ==, !=, >=, <=, <, >.

Numerički literali koriste iste oznake kao u C-jeziku, npr. 5e-2 odgovara $5 \cdot 10^{-2}$.

DPIC raspolaže sa velikim brojem standardnih funkcija, a daje i mogućnost korisniku da stvara nove u obliku macro konstrukcija. Ugrađene funkcije su: **sin**(x), **cos**(x), **log**(x), **exp**(x), **sqrt**(x), **max**(x,y), **atan2**(x,y), **min**(x,y), **int**(x), **rand**() i **srand**(). Obje funkcije exp i log su po bazi 10; rand() vraća slučajni broj između [0-1). Funkcija sprintf() ponaša se kao C sprintf() funkcija s tim da prihvaća samo %, %e, %f i %g formate stringova.

Za repetitivne dijelove dijagrama ili slike koje korisik želi nacrtati najbolji put

je definiranje vlastite funkcije ili macro naredbe koja se onda za svaki dio samo poziva s različitim argumentima.
Sintaksa je **define**(‘*ime*’, ‘*tekst koji se zamjenjuje*’)

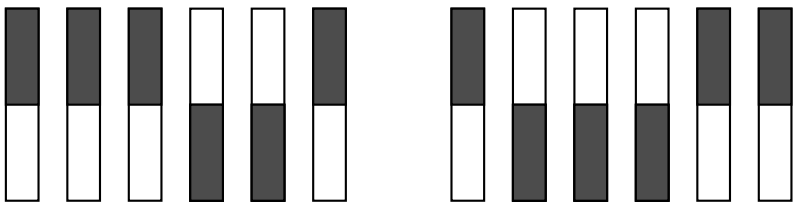
Na ovaj način definira se *ime* kao macro koji će se zamijeniti s ’*tekst-om koji se zamjenjuje*’ pri svakom pozivu, koji općenito izgleda ovako:

```
name(arg1, arg2, . . . argn)
```

Argumenti (ako se zadaju) bit će zamijenjeni za oznake **$1**, **$2** ... **$n** koje se pojavljuju u ’*tekst-u koji se zamjenjuje*’. Slijedeći primjer nacrtat će sklopke za različita stanja na ulazu prikazane slikom 2.22.

```
.PS
cct_init

sirina=0.17
razmak=0.15
raz_blk=0.4

# Macro
define(‘uklj’, ‘{
    box fill ht 0.5 wid sirina with .n at last box.c
} ’)

define(‘isklj’,‘{
    box fill ht 0.5 wid sirina with .n at last box.n
} ’)

define(‘jumper’, ‘[
  box ht 1 wid sirina ;
  if ($1)>0  then
    uklj();
  isklj();
  move right razmak
]’)
```

```
define(‘jblok’, ‘[
  jumper($1);
  jumper($2);
  jumper($3);
  jumper($4);
  jumper($5);
  jumper($6);
  move raz_blk
]’)

# Pozivanje macro-a
jblok(0,0,0,1,1,0)
jblok(0,1,1,1,0,0)
.PE
```



Slika 2.22: Primjer korištenja macro naredbi

Ako se macro **jumper()** pozove sa **jumper(1)**, vrijednost formalnog argumenta $1 bit će "**1**". Ako se pak pozove sa **jumper(veliki string)** vrijednost $1 bit će "**veliki string**".

Makro naredbe vrijede samo u dijagramu/slici koja se crta, a ne izvan nje. Poništavanje makro naredbe unutar programskog koda postiže se naredbom **undef** *ime*.

Upravljanje tijekom programa postiže se klasičnom **ako-onda-inače** naredbom, čija sintaksa izgleda ovako:

```
if izraz then X, ako je X istinit [else Y, ako je Y istinit]
```

Prvo se dakle, određuje vrijednost izraza, pa ako je ona istinita, tj. različita od nule, izvodi se odsječak X, inače se izvodi odječak Y. X i Y odsječci mogu imati jednu ili više PIC naredbi. Ako ih ima više, naredbe se trebaju staviti u vitičaste zagrade {}.

Izraz dakako može biti složen, a njegovi podizrazi odvojeni logičkim operatorima. Podržana je i usporedba stringova uz pomoć operatora == za '*je li jednako?*' i != za '*je li nejednako*'.

PIC jezik ima također i naredbu za petlju, čija je sintaksa:

```
for var = izraz1 to izraz2 [by [*]izraz3] do X
```

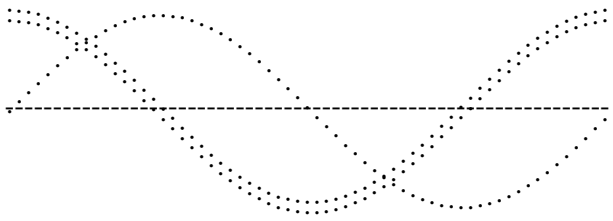
gdje je X tijelo naredbi.

Prvo se varijabla *var* postavi u vrijednost izračunatu izrazom *izraz1*. Sve dok je vrijednost varijable *var* manja ili jednaka izrazu *izraz2*, izvodi se tijelo naredbi *X* nakon čega se povečava varijabla *var* za izraz *izraz3*. Ako **by** dio nije napisan, onda je povećanje varijable *var* samo za 1. Ako izrazu *izraz3* prethodi znak ∗onda se varijabla *var* umjesto zbrajanja uvećava s umnoškom, za *izraz3* puta. Tijelo naredbi *X*, ako ih ima više, obuhvaća se parom vitičastih zagrada {}.
Slijedeći primjer nacrtat će graf funkcije *sin*() i *cos*() kako je prikazano slikom 2.23.

```
.PS
pi = atan2(0, -1);
for i = 0 to 2 * pi by 0.1 do {
    "-" at (i/2, 0);
    "." at (i/2, sin(i)/2);
    ":" at (i/2, cos(i)/2);
}
.PE
```

Na mjestima gdje se dopušta *n***th** (n-ta vrijednost) moguće je generirati i nti-izraz kao '*expr*'**th**. Pritom treba primjetiti da se oznaka **th** piše bez razmaka.
Na primjer:

```
for i = 1 to 4 do {
   line from 'i'th box.nw to 'i+1'th box.se
}
```



Slika 2.23: Primjer: graf *sin*() i *cos*() funkcije

Povezat će linijama sjeverozapadne s jugoistočnim stranama zadnjih četiriju nacrtanih okvira.

Macro dopušta uvlačenje koda iz vanjskih datoteka sa:

```
copy filename thru macro
```

ili stvaranja koda iz vlastite, s pomoću petlje.
Tako će naredbe:

```
.PS
copy thru % circle at ($1,$2) % until "END"
1 2
3 4
5 6
END
box
.PE
```

biti ekvivalentne sa:

```
.PS
circle at (1,2)
circle at (3,4)
circle at (5,6)
box
.PE
```

3 M4 preprocesor

M4 je macro procesor, koji prenosi ulaz na izlaz, izvršavajući macro naredbe tijekom procesa. Macro naredbe su već ugrađene ili definirane od samog korisnika i mogu koristiti bilo koji broj argumenata. Osim samog izvršavanja macro naredbi, M4 ima ugrađene funkcije za uključivanje datoteka, pokretanja shell naredbi, računanja cijelim brojevima, rukovanja tekstom na različite načine, izvršavanje rekurzija ... jednom riječju M4 može biti korišten kao front-end na compiler ili kao vlastiti macro procesor.

M4 program se nalazi u svakoj GNU/Linux distribuciji, i standardiziran je od strane POSIX-a[1]. Uglavnom je samo nekolicina korisnika svjesna u njegovo postojanje. Pa ipak, oni koji ga pronađu često postaju odani korisnici, neki čak ovisni. U početku ga koriste za rješavanje jednostavnih problema, a kasnije za rješavanje sve većih problema učeći kako pisati kompleksne strukture M4 macro naredbi. Na kraju predani korisnici pišu M4 aplikacije koje rješavaju sve vrste problema.

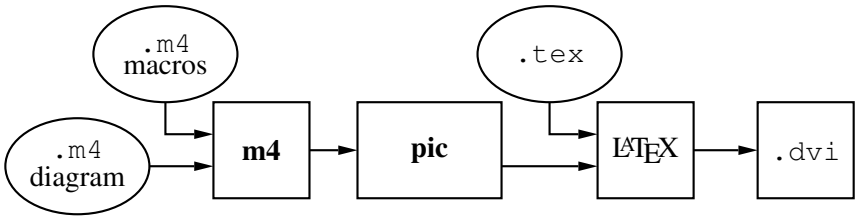
[1] Portable Operating System Interface

3.1. Macro biblioteke

M4 programski jezik je vrlo jednostavan za uporabu, što je i vidljivo iz M4 priručnika [2]. M4 dolazi s već definiranim macro naredbama. Nova macro naredba može se definirati i od strane samog korisnika. Macro je moguće kreirati i proširenjem već nekog postojećeg programa. To omogućuje da korisnik sam stvara željene macro naredbe, koje će dati željeni rezultat. Makro se definira pomoću imena i teksta kojim se ime zamjenjuje kako slijedi:

```
define(‘ime’,‘zamjenski tekst’)
```

Sama macro naredba sastoji se od dva dijela, imena i argumenata. Ime se tvori od slova, brojeva i oznake '_'. Prvi znak imena ne može biti broj. Imena macro naredbi su zavisna o velikom i malom slovu. Macro naredbu se može pozvati po njenom imenu sa ili bez argumenata (npr. *naredba(arg1, arg2, ..., argn)*).



Slika 3.1: Korištenje macro naredbi

Da bi nacrtali tehnički shemu, potrebno je izvršiti PIC program kako je prokazano dijagramom na slici 3.1. Da bi skratili vrijeme crtanja možemo koristiti već gotove definirane elemente. Skup macro naredbi za crtanje diagrama, elektroničkih i logičkih elemenata objavio je J. D. Aplevich, pod nazivom **Circuits macros**. Macro naredbe pisane su u M4, a služe se PIC programskim jezikom za crtanje grafičkih elemenata. Da bi koristili te već gotove elemente (npr. *otpornik*), potrebno je prilikom kompajliranja sheme uključiti odgovarajuće datoteke koje sadrže korištene macro naredbe.

U ovisnosti o tome kakav izlaz želimo, i kojim PIC programom (*dpic*, *gpic*) želimo izvršiti konverziju, razlikuju se datoteke koje se uključuju.

Da bi se konvertirala shema pomoću *dpic* i *PSTrick* potrebno je izvršiti sljedeće naredbe:

```
m4 pstricks.m4 libcct.m4 shema.m4 > shema.pic
dpic -p shema.pic > shema.tex
```

Da bi tako dobivenu *tex* datoteku uključili u LaTeX dokument, potrebno je u zaglavlje LaTeX dokumenta dodati:

```
\usepackage{pstricks}
```

a na mjestu gdje se želi nacrtati shemu:

```
\begin{figure}[hbt]
    \centering
    \input shema.tex
    \caption{Opis slike}
    \label{shema}
\end{figure}
```

Također, želi li se pretvorbu izvršiti za *Tikz PGF*, to se može učiniti na sljedeći način:

```
m4 pgf.m4 libcct.m4 shema.m4 > shema.pic
dpic -g shema.pic > shema.tex
```

Da bi tako dobivenu *tex* datoteku uključili u LaTeX dokument, potrebno je u zaglavlje LaTeX dokumenta dodati:

```
\usepackage{tikz}
```

a na mjestu gdje želimo nacrtati shemu isti kôd kao i za PSTricks paket.

Konverziju možemo vršiti i pomoću *gpic* programa. Sljedeći primjer opisuje način kako se to čini koristeći *PSTrick*:

```
m4 pstricks.m4 libcct.m4 shema.m4 > shema.pic
gpic -t shema.pic > shema.tex
```

ili *Tikz PGF*

```
m4 pgf.m4 libcct.m4 shema.m4 > shema.pic
gpic -t shema.pic > shema.tex
```

Tako dobivena LaTeX datoteka pretvara se s pomoću LaTeX ili *PDFlatex* programa, što kao izlaz daje DVI ili PDF dokument kako je prikazano dijagramom na slici 3.1. Tako dobiveni dokument kasnije je moguće pretvoriti u neki drugi grafički format (npr. PNG, EPS).

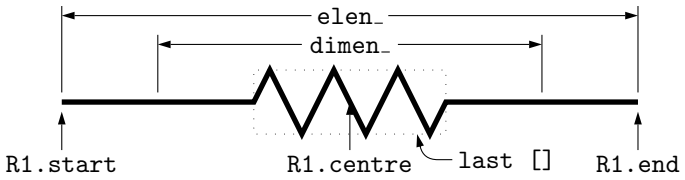
3.2. M4 macro-i za električne krugove

Kako je na mojoj katedri učestalo crtanje grafičkih elemenata iz područja elektrotehnike, automatske regulacije, računalstva, obrađeni su elementi koji će olakšati kreiranje tehničkih shema iz tih područja, što će pridonjeti kvalitetnijoj dokumentaciji.

3.2.1. Osnove električnih elemenata i krugova

Kako svi električni elementi koji će biti obrađeni u ovom poglavlju imaju predefinirane vrijednosti, nije ih zasebno definirati. Moguće je njihovo mijenjanje ukoliko je potrebno. Elementi su definirani u paketu *Circuit macros* i slobodni su za preuzimanje i korištenje.

Na primjeru otpornika prikazanog na slici 3.2 dat je kratki prikaz osnovnih veličina i predefiniranih pozicija.



Slika 3.2: Osnovna shema otpornika R1

Da bi nacrtali otpornik R1 prikazan slikom 3.2 korišten je programski kod:

```
.PS
cct_init
linewid = 2.0
R1: resistor
.PE
```

gdje *cct_init* poziva gotovo prazni macro koji postavlja vrijednosti lokalnih varijabli potrebnih za crtanje nekih elemenata. Taj makro moguće je mjenjati da bi se postavile željene vrijednosti. Tako je moguće npr. promjeniti veličinu elementa pomoću naredbe *linewid = 2.0* što odgovara veličini elementa od 3 incha a prikazano je kotom *elen_*. Kota *dimen_* označava veličinu samog otpornika.

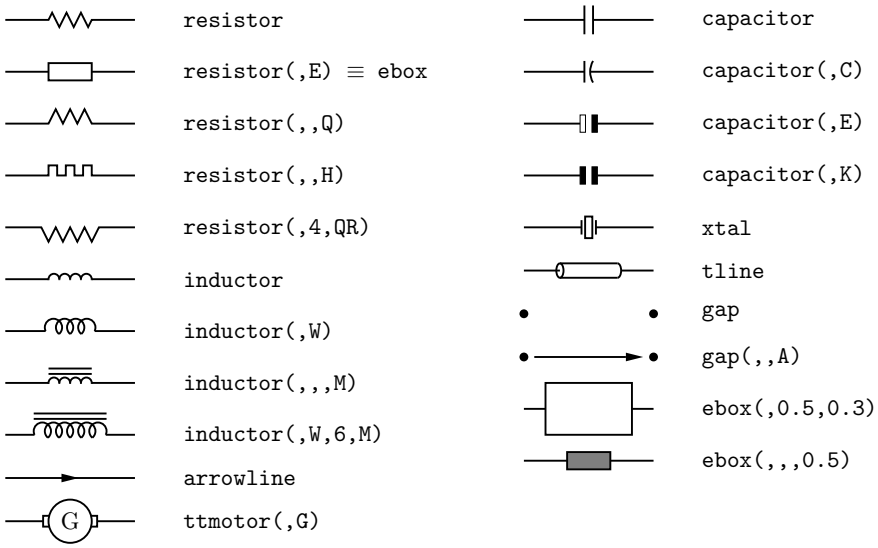
Pozicije elementa R1 označene su sa *R1.start*, *R1.center* i *R1.end*, a omogućuju nam da element R1 povežemo s drugim elementima u shemi.

Gotovi svi električni elementi, međusobno su povezani linijama i strelicama koje označavaju tok informacije ili električne energije.

3.2.2. Elementi s dva priključka

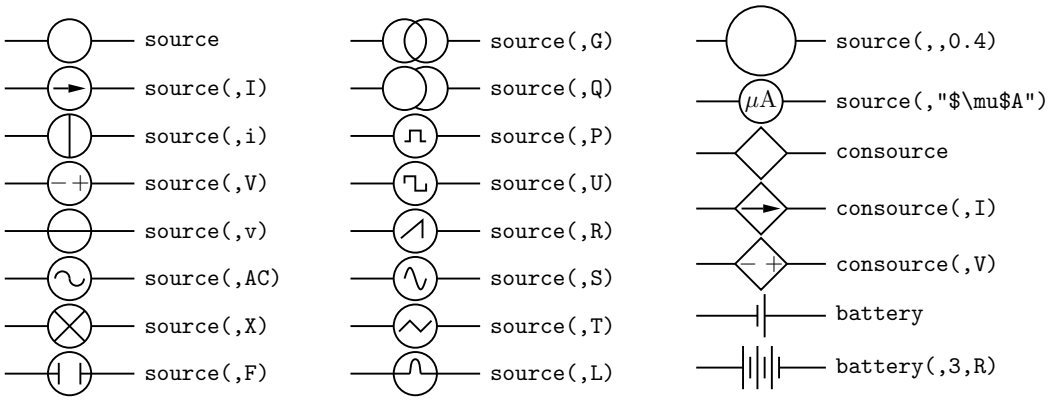
Najčešće korišteni elementi u crtanju električnih shema su elementi s dva priključka. Takvi elementi postoje u velikom broju izvedbi. U ovom poglavlju obrađeni su samo često korišteni dok se ostali na lak način mogu načiniti od strane korisnika, i to na način da se već gotovom osnovnom elementu dodaju neke opcije.

Makro naredbe koje služe za crtanje otpornika, induktora, kondenzatora prikazane su na slici 3.3.



Slika 3.3: Macro naredbe: Otpornici, induktori, kondenzatori

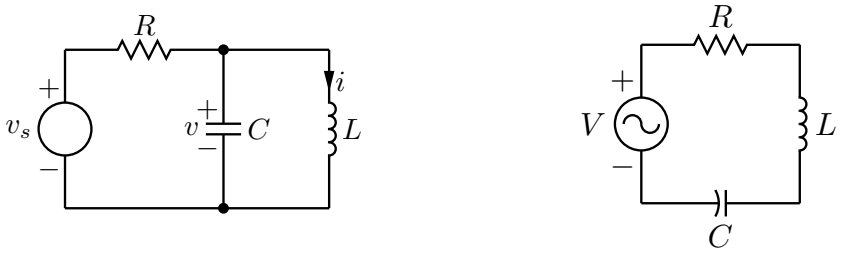
Makro naredbe koje služe za crtanje raznih tipova električnih izvora prikazane su na slici 3.4.



Slika 3.4: Macro naredbe: Tipovi električnih izvori

Dati primjer crta RLC spoj prikazan slikom 3.5a.

```
.PS
cct_init

elen = 0.75
Origin: Here
   source(up_ elen); llabel(-,v_s,+)
   resistor(right_ elen);  llabel(,R,)
   dot
   {
      capacitor(down_ to (Here,Origin))
      rlabel(+,v,-); llabel(,C,)
      dot
      }
   line right_ elen*2/3
   inductor(down_ Here.y-Origin.y); llabel(,L,); b_current(i)
   line to Origin
.PE
```



Slika 3.5: RLC spojevi

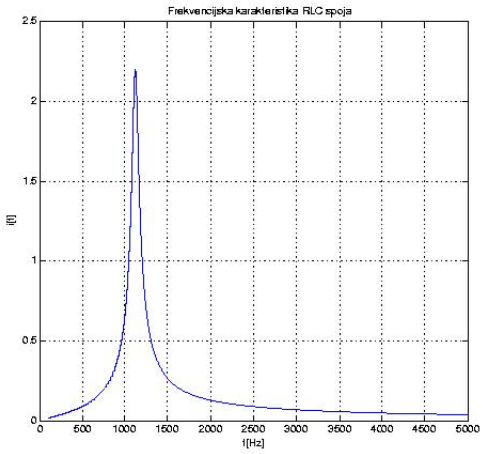
Slijedi interaktivni primjer RLC spoja prikazanog slikom 3.5b, a opisan jednadžbom:

$$i = \frac{V}{\sqrt{R^2 + \left[(2\pi f L)^2 - \left(\frac{1}{2\pi f C}\right)^2\right]}}$$

Promjenom parametara mijenja se i frekvencijska karakteristika RLC spoja.

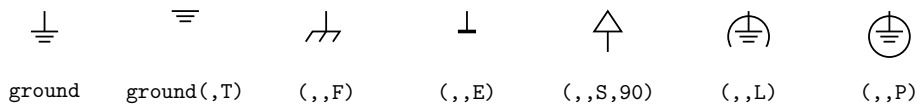
R	=	100	Ω	
L	=	0.2	H	
C	=	0.1	μF	
f_{poc}	=	100	Hz	
f_{kr}	=	5000	Hz	izvrsi

Primjer grafa dobivenog nakon izvršavanja primjera prikazuje slika 3.6.



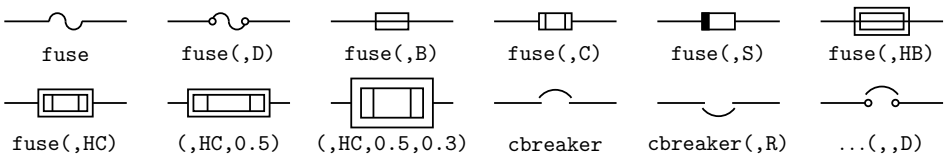
Slika 3.6: Frekvencijska karakteristika RLC spoja

Makro naredbe za crtanje raznih tipova uzemljenja prikazane su na slici 3.7.



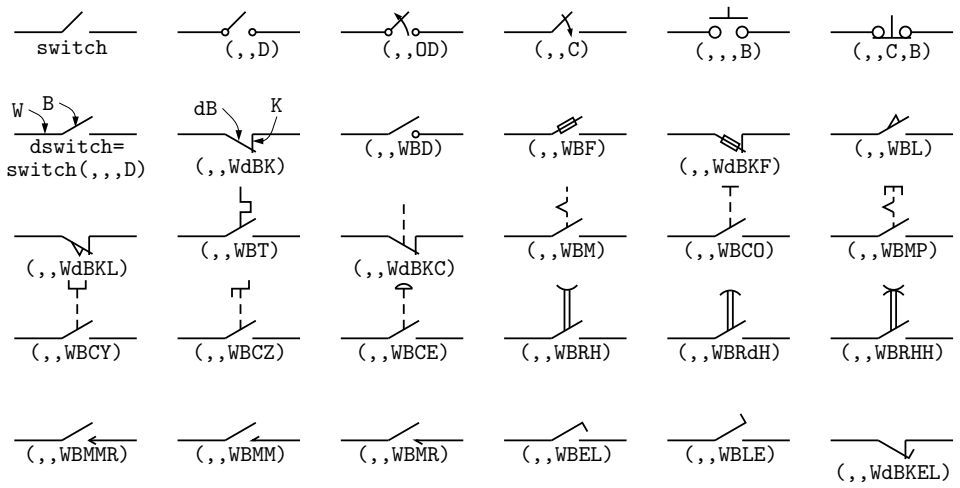
Slika 3.7: Macro naredbe: Uzemljenja

Makro naredbe za crtanje raznih tipova osigurača prikazuje slika 3.8.



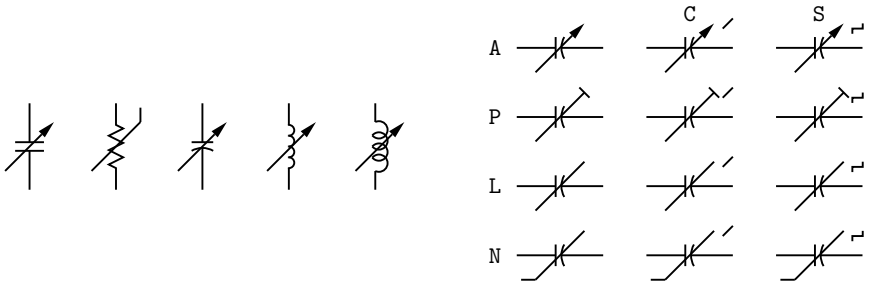
Slika 3.8: Macro naredbe: Osigurači

Makro naredbe za crtanje raznih tipova sklopki prikazuje slika 3.9.



Slika 3.9: Macro naredbe: Sklopke

Želimo li element načiniti promjenljivim, to nam je omogućeno korištenjem macroa *variable* i neki od argumenata prikazanih slikom 3.10. Slijedeći primjer



Slika 3.10: Macro naredbe: *variable*

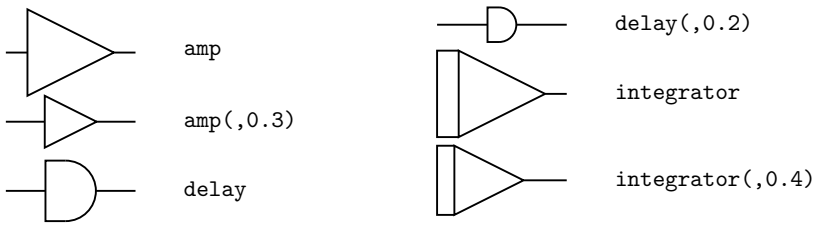
nacrtat će kondenzator precrtan strelicom tipa A.

```
.PS
cct_init
variable(capacitor, A)
.PE
```

3.2.3. Analogna i digitalna elektronika

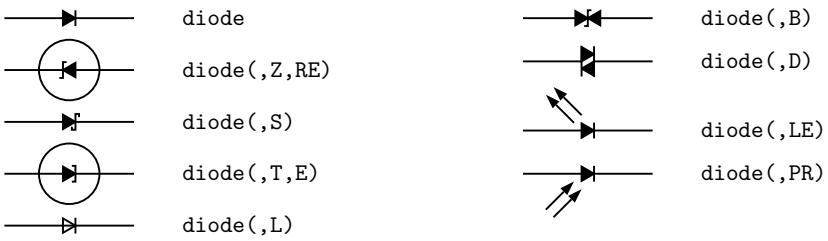
Slijede macro naredbe za crtanje elektroničkih elemenata.

Makro naredbe za crtanje raznih tipova pojačala i integratora prikazane su na slici 3.11.



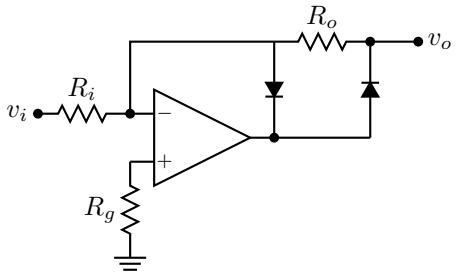
Slika 3.11: Macro naredbe: Pojačala, integratori

Makro naredbe za crtanje raznih tipova dioda prikazane su na slici 3.12.



Slika 3.12: Macro naredbe: Tipovi dioda

Želi li se nacrtati shema električnog kruga prikazana slikom 3.13, potrebno je

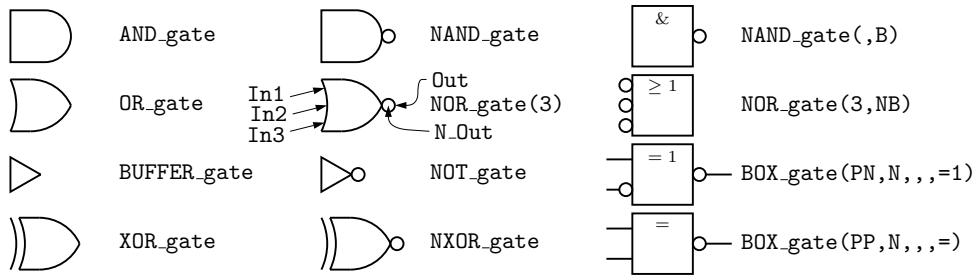


Slika 3.13: Primjer električnog kruga

napisati sljedeći program:

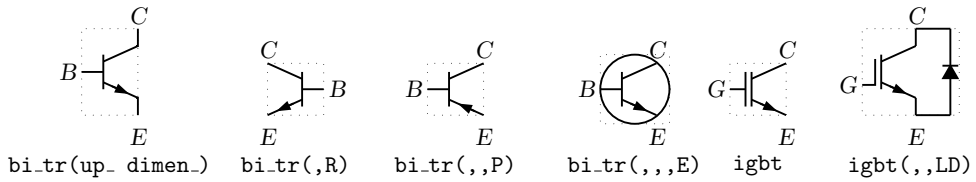
```
.PS
cct_init
   dot; "$v_i$" rjust at last [].w
   resistor(right_ dimen_) ; llabel(,R_i)

T: dot
   line right_ linewid/4

A: opamp with .In1 at Here
   line from A.In2 to (T,A.In2)
   resistor(down_ dimen_) ; rlabel(,R_g)
   ground(,T)
   line right_ dimen_ from A.Out
   diode(up_ dimen_); dot
   {line right_ linewid/2; dot; "$v_o$" ljust at last [].e }
   resistor(left_ to (A.Out,Here)) ; rlabel(,R_o)
   { diode(down_ to (Here,A.Out)); dot }
   line to (T,Here) then to T
.PE
```

Makro naredbe za crtanje raznih tipova logičkih vrata prikazane su slikom 3.14.



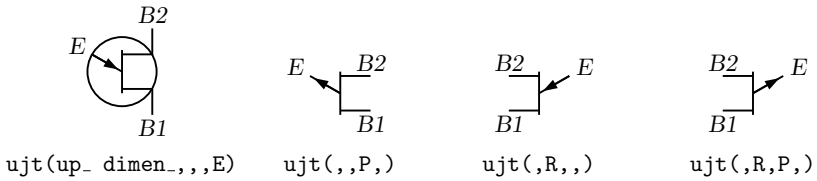
Slika 3.14: Macro naredbe: Logička vrata

Makro naredbe za crtanje raznih tipova bipolarnih tranzistora prikazane su slikom 3.15. Smjer crtanja je prema gore.



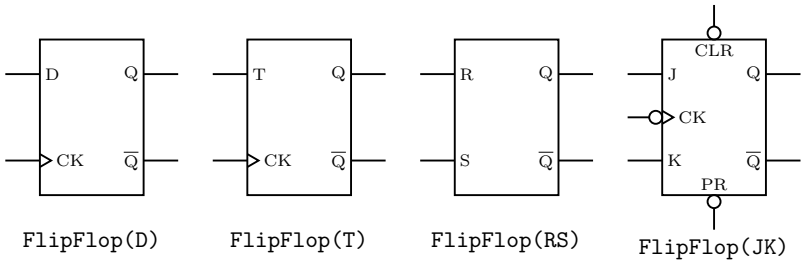
Slika 3.15: Macro naredbe: Bipolarni tranzistori

Makro naredbe za crtanje raznih tipova UJT elemenata prikazane su slikom 3.16.



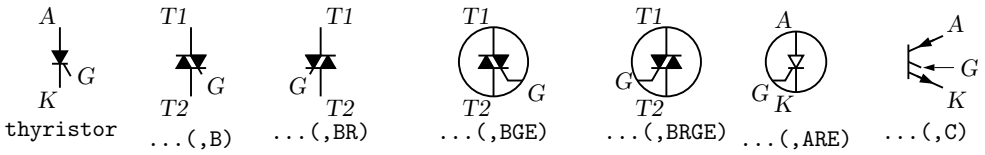
Slika 3.16: Macro naredbe: UJT elementi

Makro naredbe za crtanje raznih Flip Flop elemenata prikazane su slikom 3.17.



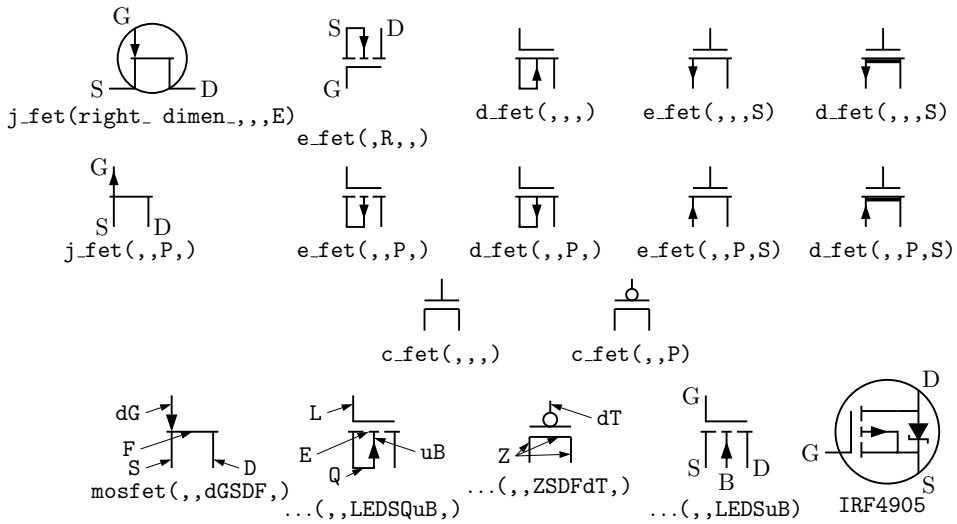
Slika 3.17: Macro naredbe: Flip Flop elementi

Makro naredbe za crtanje raznih tipova tiristora prikazane su slikom 3.18.



Slika 3.18: Macro naredbe: Tiristori

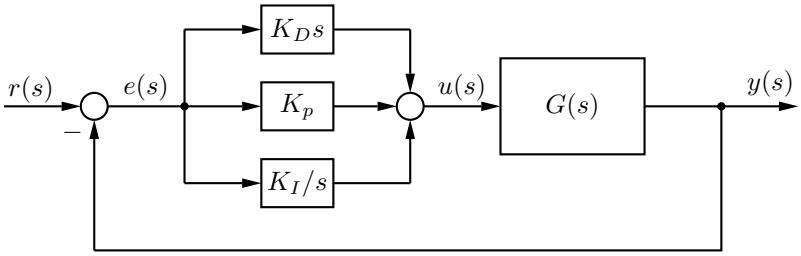
Makro naredbe za crtanje raznih tipova FET elemenata prikazane su slikom 3.19.



Slika 3.19: Macro naredbe: FET elementi

3.3. Crtanje blokova automatske regulacije

Blokovi automatske regulacije općenito se sastoje od 3 različita elemenata i to: točke grananja, točke zbrajanja i bloka. Slikom 3.20 prikazan je PID controler, i program koji omogućava njegovo crtanje.

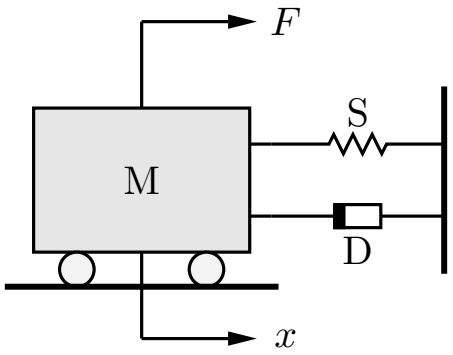


Slika 3.20: PID kontroler

```
.PS
cct_init
  linewid = linewid*0.8
  circlerad = 0.07
  bw = boxwid/2
  bh = boxht/2

PID: [
  {"$r(s)$" above ljust}
  arrow
S1: circle
  line right "$e(s)$" above
  { arrow right ; box ht bh wid bw "$K_p$"
    arrow right linewid-circlerad ; S2: circle }
  { line up linewid ; arrow right ; box ht bh wid bw "$K_D s$"
    line to (S2,Here) ; arrow to S2.n }
  { line down linewid ; arrow right ; box ht bh wid bw "$K_I/s$"
    line to (S2,Here) ; arrow to S2.s }
  { dot }
```

```
  arrow right from S2.e "$u(s)$" above
  box "$G(s)$"
  line right; dot;
  arrow right; "$y(s)$" above rjust at Here+(0,2pt__);
  line down boxht*3/2 from last line.end then left last line.end.x-S1.x
  arrow to S1.s
  "$-\;$" below rjust
  ]
.PE
```

Kako je vidljivo iz programa za crtanje točke grananja koristi se naredba *dot*. Za crtanje točke zbrajanja koristi se naredba *circle*, kojoj je radijus degiriran naredbom *circlerad = 0.07*. Naredba *box* nacrtat će blok. Sve je međusobno povezano linijama (naredba *line*) i stelicama (naredba *arrow*). Unutar bloka moguće je napisati bilo koju jednadžbu koristeći LaTeX.



Slika 3.21: Mehanički MDS sustav

Uzmemo li primjer mehaničkog sustava s masom M, oprugom S i prigušivačem D, uz narinutu vanjsku silu F kako je prikazano slikom 3.21. Sustav je opisan sljedećom jednadžbom:

$$M\ddot{x} + D\dot{x} + Sx = F$$

Jednadžba opisuje sustav drugog reda koji ima dva spremnika energije koji su međusobno nezavisni: *masu* kao spremnik kinetičke energije i *oprugu* kao spremnik potencijalne energije.

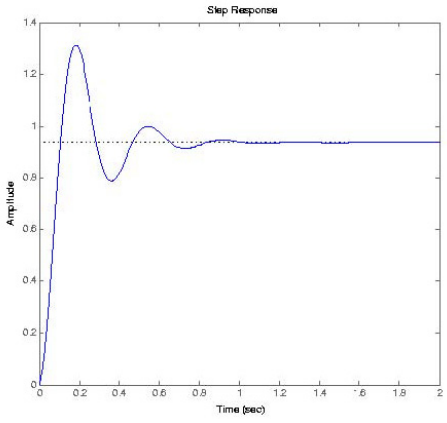
Sljedeći interaktivni primjer omogućuje nam da za MDS sustav odabiremo proizvoljne parametre, te nakon klika na gumb *izvrši* kao izlaz dobivamo graf, zavisan o zadanim parametrima i regulatoru koji se koristi.

M = 1 kg
D = 10 kg/s
S = 20 kg/s^2
F = 1 N
K_P = 300
K_D = 10
K_I = 70

- otvoreni sustav
- P regulator
- PD regulator
- PI regulator
- PID regulator

izvrsi

Primjer grafa dobivenog nakon izvršavanja primjera s P regulatorom prikazuje slika 3.22.



Slika 3.22: Mehanički MDS sustav za P regulator

3.4. Crtanje progamskih dijagrama toka

Programski diagram toka podataka je grafički prikaz algoritma. Zapisivanje se vrši međunarodno dogovorenim simbolima i ne ovisi o govornom jeziku onoga koji sastavlja algoritam. Grafički prikaz je jednostavan, pregledan, lako se pronalaze greške. Nadalje, problem se može jednostavno analizirati, usporediti s nekim drugim problemom, skratiti vrijeme pronalaženja rješenja.

Macro naredbe koje opisuju pojedine dijelove dijagrama toka:

```
define(‘fboxwid’,linewid*2)
define(‘fboxht’,linewid*0.75)
define(‘farrowht’,linewid*0.5)

# početak
define(‘fstart’,‘[B:ellipse height 0.3 \
  fill_(fillval) ‘$1’
  N: B.n; S:B.s; E:B.e; W: B.w
  ]’)
# kraj
define(‘fend’,‘[B:ellipse height 0.3 \
  fill_(fillval) ‘$1’
  N: B.n; S:B.s; E:B.e; W: B.w
  ]’)
# deklaracija varijabli i konstanti
# postavljanje na početnu vrijednost
# obrada podataka
define(‘fbox’,‘[B:box ifelse(‘$2’,,‘wid fboxwid ht fboxht’,‘$2’) \
  fill_(fillval) ‘$1’
  N: B.n; S:B.s; E:B.e; W: B.w
  ]’)
# ulaz podataka
# izlaz podataka
define(‘fdata’,‘[B:box ifelse(‘$2’,,‘wid fboxwid*0.8 ht \
  fboxht*1.2’,‘$2’) invis
```

```
  N: B.n; S: B.s; E: B.e; W: B.w
  shade(fillval,
   L1: line from B.sw to B.nw+(fboxwid*0.1, 0);
   L2: line from B.ne+(fboxwid*0.1, 0) to B.se;
   line from L1.start to L2.end
  )
  ifelse(‘$1’,,,‘$1’ at B.c)
  ]’)
# grananje
define(‘ftest’,‘[B:box ifelse(‘$2’,,‘wid fboxwid*0.8 ht \
  fboxht*1.2’,‘$2’) invis
  N: B.n; S: B.s; E: B.e; W: B.w
  shade(fillval,line from 0.5 between last box.n and last box.e \
    to last box.e then to last box.s then to last box.w then to \
    last box.n \
    then to 0.5 between last box.n and last box.e)
  ifelse(‘$1’,,,‘$1’ at B)
  ]’)
# petlje
define(‘fcase’,‘[ down; S: Here; xe = S.x
  fcaseloop_(1,$@)
  ifelse(‘$2’,,‘E:S; W:S; N:S’,‘E:(xe+linewid/2,B1.E.y); W:T1.W; \
  N:T1.N’)
  fcasearrow_(1,$@)
  arrow from E to (E,S) then to S ]’)
  define(‘fcaseloop_’,‘ifelse(‘$3’,,,
   ‘T‘$1’: ftest(‘$3’,‘$2’) with .N at S
    B‘$1’: ‘$4’ with .nw at T‘$1’.E+(linewid/2,min(T‘$1’.ht, \
      fboxht)/2)
      arrow right linewid/4 from T‘$1’.E then down \
      T‘$1’.E.y-B‘$1’.W.y then to B‘$1’.W
    S: (T‘$1’.S.x,min(T‘$1’.S.y,B‘$1’.S.y)-linewid/3)
      ifelse(‘$5’,,‘line’,‘arrow’) from T‘$1’.S to S
```

```
      xe = max(xe,B‘$1’.E.x)
      fcaseloop_(incr($1),‘$2’,shift(shift(shift(shift($@)))))’)’)
  define(‘fcasearrow_’,‘ifelse(‘$3’,,,
   ‘arrow from B‘$1’.E to (E,B‘$1’.E)
    fcasearrow_(incr($1),‘$2’,shift(shift(shift(shift($@)))))’)’)
define(‘fwhiledo’,‘[ down
  T: ftest(‘$1’,‘$2’)
    ifelse(‘$4’,,"T",‘$4’) above ljust at T.E
    arrow right linewid/2 from T.E
  B: ‘$3’ with .W at Here
  E: B.E; W: T.w; S: T.S
    arrow up max(linewid/4,T.n.y-B.n.y+arrowht*1.5) from B.N then \
    left B.x-T.x
  N: Here
    arrow to T.n ]’)
define(‘frepeatuntil’,‘[
  N: Here
  B: ‘$3’ with .N at N
  W: B.W
    arrow down linewid/3
  T: ftest(‘$1’,‘$2’)
  E: B.E+(linewid/2+max(0,T.e.x-B.e.x),0)
    arrow from T.e to (E,T) then to E then to B.E
    ifelse(‘$4’,,"F",‘$4’) above ljust at T.e
  S: T.S ]’)
define(‘fifthenelse’,‘[
  T: ftest(‘$1’,‘$2’)
  N: T.N
    ifelse(‘$5’,,"F",‘$5’) below rjust at T.W
    ifelse(‘$6’,,"T",‘$6’) below ljust at T.E
  L: ifelse(‘$3’,,‘T.S; W:T.W; LS:L’,‘$3 with .ne at \
    ((T.W.x+T.x)/2,T.S.y)
    W: L.W; LS:L.S
```

```
    arrow from T.W to (L.N,T.W) then to L.N')
  R: ifelse('$4',,'T.S; E:T.E; RS:R','$4 with .nw at \
    ((T.E.x+T.x)/2,T.S.y)
    E: R.E; RS: R.S
    arrow from T.E to (R.N,T.E) then to R.N')
  S: (T.x,min(LS.y,RS.y)-linewid/3)
    arrow from LS to (LS,S)
    arrow from RS to (RS,S)
    line to (LS,Here)
  ]')
# spojna točka
define('fdot','[B:circle rad 0.1 \
  fill_(fillval)
  N: B.n; S:B.s; E:B.e; W: B.w
  ]')
```

Osnovni simboli dijagrama toka i njihove macro naredbe su:

- početak - *fstart*
- ulaz podataka - *fdata*
- deklaracija varijabli i konstanti; postavljanje na početnu vrijednost; obrada podataka - *fbox*
- izlaz podataka - *fdata*
- kraj - *fend*
- spojna točka - *fdot*

Simboli za ulaz i izlaz podataka su istog oblika. Kod jednostavnih algoritama ulaz i izlaz su odmah uočljivi. Simboli dijagrama toka se povezuju strelicama koje pokazuju tok podataka.

Simbol za grananje je romb, naredba *ftest*, a odgovori na postavljeni logički uvjet mogu biti **DA** i **NE**, odnosno **T** i **F** (*true* i *false*).

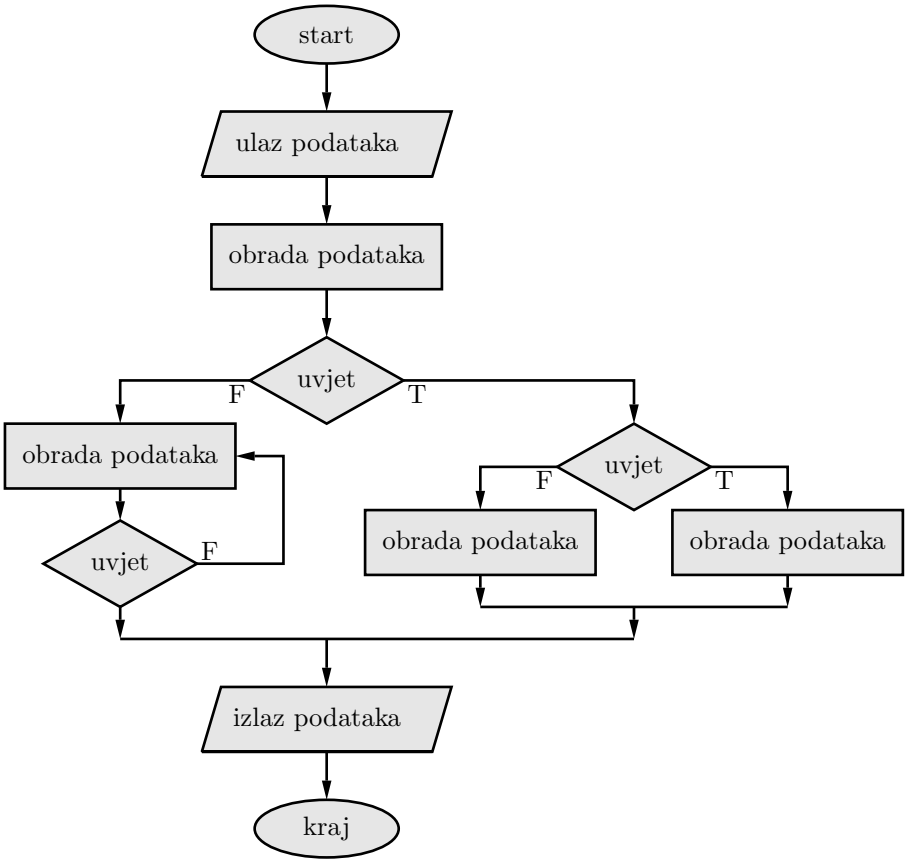
Dio programa, niz istih naredbi koje se ponavljaju dok je neki uvjet zadovoljen ili dok ne postane zadovoljen, naziva se petlja. Postoji nekoliko vrsti petlji.
Date su macro naredbe za sljedeće petlje:

- case - *fcase*
- while do - *fwhiledo*
- repeat until - *frepeatuntil*
- it then else - *fifthenelse*

Koristeći se gore navedenim makro naredbama, dat je primjer diagrama toka. Primjer je prikazan slikom 3.23.

```
.PS
  linethick_(1.0)
  arrowwid = 0.05
  arrowht = 0.1
  fillval = 0.9

  down
  fstart("start")
  arrow linewid/2
  fdata("ulaz podataka", wid fboxwid*1.2 ht fboxht*.9)
  arrow linewid/2
  fbox("obrada podataka", wid fboxwid*1.2 ht fboxht*.9)
  arrow down linewid/2 from last [].S

  fifthenelse("uvjet",,
    frepeatuntil("uvjet",,fbox("obrada podataka", \
    wid fboxwid*1.2 ht fboxht*.9)),
    fifthenelse("uvjet",,
      fbox("obrada podataka", wid fboxwid*1.2 ht fboxht*.9),
      fbox("obrada podataka", wid fboxwid*1.2 ht fboxht*.9)dnl
      )dnl
```

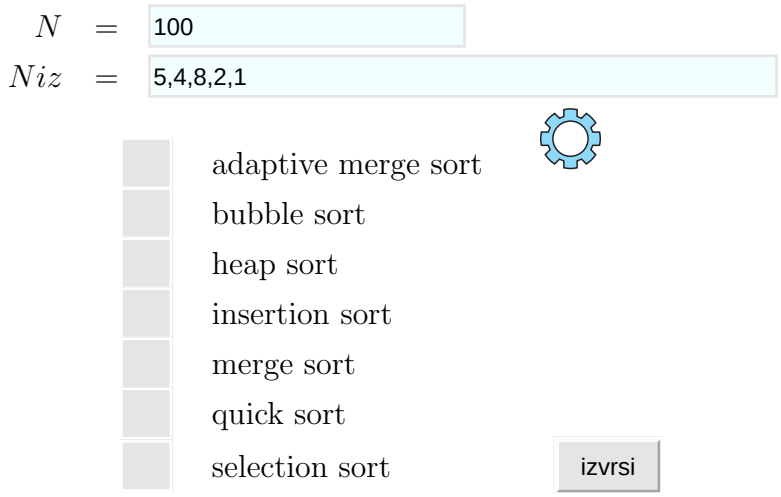
```
    ) with .N at Here

  arrow down 0.25 from last [].S
  fdata("izlaz podataka", wid fboxwid*1.2 ht fboxht*.9)
  arrow linewid/2
  fend("kraj")
.PE
```



Slika 3.23: Diagram toka

Simboli korišteni za crtanje diagrama toka, također se mogu koristiti i za crtanje drugih diagrama i shema. Moguće je i napisat macro naredbe za druge elemente koji još nisu definirani.

Sljedeći interaktivni primjer sortira zadani niz cijelih brojeva za odabrane vrste sortiranja.



Primjer dobivenog izlaza nakon izvršavanja primjera s $N = 100$ slučajnih brojeva, za odabrane sve vrste sortiranja:

```
Ne sortirana lista:
96  65  10  74  84  68  85  25  36  71
40  86  26  66  76  58  36  99  86  84
85  40  41  43  32  81  17  10  11  16
21  62  15  53  16  66  10  62  55  75
15  56  55  10  65  24  38  32  38  92
22  10  24  89  53  69  83  58  93  9
3  75  73  75  30  53  3  89  76  18
17  23  26  77  92  84  44  36  55  8
19  17  98  33  60  51  46  25  98  96
18  96  98  15  51  71  16  18  58  17

adaptive_merge_sort traje 0.000ms
bubble_sort traje 0.000ms
```

```
heap_sort traje 0.000ms
insertion_sort traje 0.000ms
merge_sort traje 0.000ms
quick_sort traje 0.000ms
selection_sort traje 10.000ms

Sortirana lista:
3  3  8  9  10  10  10  10  10  11
15  15  15  16  16  16  17  17  17  17
18  18  18  19  21  22  23  24  24  25
25  26  26  30  32  32  33  36  36  36
38  38  40  40  41  43  44  46  51  51
53  53  53  55  55  55  56  58  58  58
60  62  62  65  65  66  66  68  69  71
71  73  74  75  75  75  76  76  77  81
83  84  84  84  85  85  86  86  89  89
92  92  93  96  96  96  98  98  98  99
```

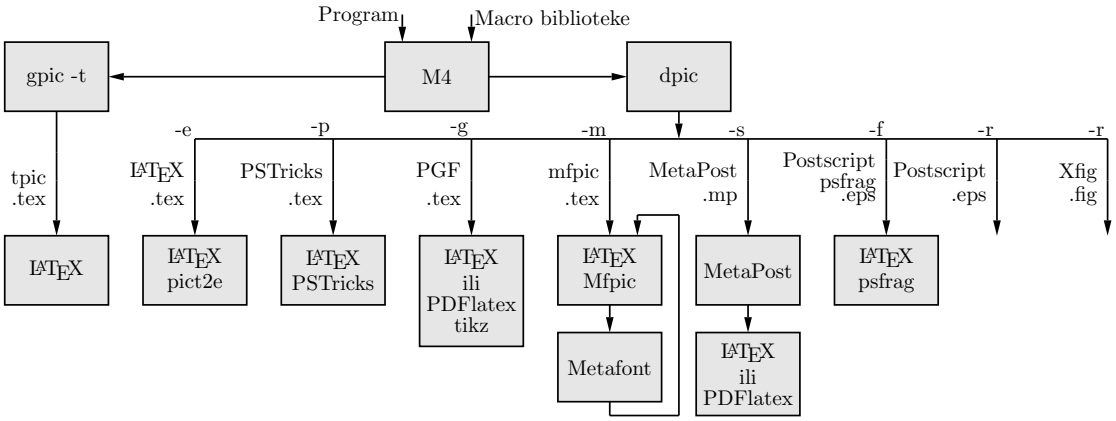
U slučaju zadavanja niza ulaznih podataka, program izvršava odabrane metode sortiranja nad tim podacima, ne uzimajući pritom u obzir varijablu N koja se odnosi na polje slučajnih podataka. Ako se želi obrada N slučajnih podataka, onda polje uz varijablu *Niz* mora ostaviti prazno.

3.5. Alternativni izlazni formati

Kako je prikazano, koristeći se raznim macro naredbama moguće je crtanje raznih tehničkih shema i dijagrama. Tako nacrtane sheme potrebno je pospremiti u neki format, da bi se mogau uključiti u željeni dokument. Vrlo važan podatak je da pozivanje izvršitelja *gpic -t* tvori izlaz koji uključuje *tpic* *special* naredbe, koje moraju biti konvertirane u željeni izlaz dok *dpic* tvori nekoliko zamjenskih formata kako je prikazano dijagramom na slici 3.24. Najlakša metoda tvorbe web dokumenta je tvorba postscript dokumenta, i njegova pretvorba u pdf dokument pomoću Adobe Distiller ili nekog drugog programa. Ukoliko želimo odmah načiniti pdf dokument, bez prethodno načinjenog postscript dokumenta, to je

moguće pomoću *PDFlatex* programa, ali kako *PDFlatex* ne podržava *tpic* *special* naredbe, potrebno je koristiti *dpic* za pretvorbu.

Ovisno o distribuciji *PSTricks* programi, često se nemogu uspješno izvršiti pomoću *PDFlatex* izvršitelja. Međutim, *TikZ PGF* dokument kojeg tvori *dpic* je kompatibilan i s LaTeX i s *PDFlatex*. Nekoliko zamjenskih formata koje tvori *dpic* kao što je *mfpic* i *MetaPost* također se mogu uzeti u obzir.



Slika 3.24: Izlazni formati dobiveni pomoću *gpic -t* i *dpic*

Diagram prikazan slikom 3.24 nacrtan je programom koji slijedi:

```
.PS 7.2
  cct_init
  fillval = 0.9
B1: box "gpic -t" fill_(fillval)
  line down 0.3 from B1.s;
  line down 0.2 "tpic " rjust below;
  arrow down 0.4 ".tex " above rjust;
  box "\LaTeX" fill_(fillval)
  line <- right 2 from B1.e
B2: box "M4" fill_(fillval)
  line <- up 0.2 from B2.n-(0.25,0) "Program" rjust above
  line <- up 0.2 from B2.n+(0.25,0) "Macro biblioteke" ljust above
  line -> right 1 from B2.e
B3: box "dpic" fill_(fillval)
```

```
L1: line -> down 0.2 from B3.s
L2: line left from L1.end; "-m" rjust above;
  line down 0.3 "mfpic " rjust below;
  arrow down 0.4 ".tex " above rjust;
  box "\LaTeX" "Mfpic" fill_(fillval); arrow down 0.2;
  box "Metafont" fill_(fillval); line down 0.1;
  line right 0.5; line up 1.45; line left 0.3; arrow down 0.15;
L3: line left 1 from L2.end; "-g" rjust above;
  line down 0.3 "PGF " rjust below; arrow down 0.4 ".tex " \
  above rjust; box height 0.8 "\LaTeX" "ili" "PDFlatex" \
  "tikz" fill_(fillval);
L4: line left 1 from L3.end; "-p" rjust above;
  line down 0.3 "PSTricks " rjust below; arrow down 0.4 ".tex " \
  above rjust; box "\LaTeX" "PSTricks" fill_(fillval);
L5: line left 1 from L4.end; "-e" rjust above;
  line down 0.3 "\LaTeX\ " rjust below; arrow down 0.4 ".tex " \
  above rjust; box "\LaTeX" "pict2e" fill_(fillval);
L6: line right from L1.end; "-s" rjust above;
  line down 0.3 "MetaPost " rjust below; arrow down 0.4 ".mp " \
  above rjust; box "MetaPost" fill_(fillval); arrow down 0.2;
  box height 0.6 "\LaTeX" "ili" "PDFlatex" fill_(fillval);
L7: line right 1 from L6.end; "-f" rjust above;
  line down 0.15 "Postscript " rjust below;
  line down 0.15 "psfrag " rjust below; arrow down 0.4 ".eps " \
  above rjust; box "\LaTeX" "psfrag" fill_(fillval);
L8: line right 0.8 from L7.end; "-r" rjust above;
  line down 0.3 "Postscript " rjust below; arrow down 0.4 ".eps " \
  above rjust;
L9: line right 0.8 from L8.end; "-r" rjust above;
  line down 0.3 "Xfig " rjust below; arrow down 0.4 ".fig " \
  above rjust;
.PE
```

4 Crtanje tehničkih slika u Scriptrunner sustavu

4.1. Scriptrunner

Scriptruner je web aplikacija koja omogućuje korisniku izvršavanje raznih programskih jezika on-line, pohranu podataka i izvođenje raznih modula. Njezin primarni cilj je pomoći studentima da nauče programirati. Trenutno se koristi u održavanju nastave u kolegijima Algoritamske tehnike, Računalna matematika i Web programiranje na fakultetu Strojarstva i brodogradnje Sveučilišta u Zagrebu.

Slika 4.1: Scriptruner 4

O Scriptrunneru se izlagalo na raznim konferencijama i privukao je pažnju mnogih sveučilišnih profesora diljem svijeta. U jeseni 2003. godine Scriptrunner se koristi i na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu. Od tada se koristi u nastavi Prirodoslovno matematičkog fakulteta Sveučilišta u Zagrebu i Matematičkog Odjela Sveučilišta u Osijeku.

Scriptrunner je dio projekta "Web kolaboracijski sustav". Scriptrunner kao takav je serverska aplikacija koja dozvoljava spajanje ovlaštenih korisnika putem

http protokola, koristeći neki od web browsera. Također korisnik može izvršavati razne programe pisane u nekom od programskih jezika: C, C++, Fortran, Java, LaTeX, HTML, Pascal, PHP, PIC, Python, Octava, M4, Matlab ...

Korisnici mogu razmjenjivati datoteke i organizirati ih u mape. Također svaki korisnik može mjenjati odgovarajuće postavke. Administratori sustava mogu dodavati i brisati korisnike, te postavljati ovlasti za korištenje pojednih modula.

4.1.1. Korisničko sučelje

Svaki korisnik mora se prilikom prijave autorizirati s lozinkom i korisničkim imenom. Nakon ispravnog prijavljivanja, korisniku se otvara korisničko sučelje kao što prikazuje slika 4.2. Ekran je podijeljen na četiri cjeline (eng. *frame*).

Slika 4.2: Korisničko sučelje

Tree frame

U gornjem lijevom okviru nalazi se stablo mapa (eng. folder) koje se sastoji od četiri osnovne mape ili direktorija.

- *My Files* je mapa koja može sadržavati proizvoljan proj podmapa. Svaki korisnik ima svoju My Files mapu i u njoj podmape i datoteke koje su vidljive samo njemu, vlasniku. Korisnik je slobodan kako će urediti svoju My Files mapu tj. kako će organizirati svoje podmape, datoteke i tekstove unutar nje.

- *Public files* mapa je otvorena svima tj. svi korisnici koji se mogu prijaviti na servis imaju pristup ovoj mapi. Sadržaj Public files mape određuje samo korisnik s administratorskim ovlastima i jedino on ima pravo promjene sadržaja, dok svi ostali korisnici mogu čitati sadržaje ili pokretati programe. Ova mapa namijenjena je zajedničkim sadržajima, primjerima, knjigama, skriptama i sl. Svaki korisnik može prijaviti određeni sadržaj za Public files mapu. Administrator će utvrditi je li predloženi sadržaj zanimljiv za sve korisnike i ako jest, učiniti da svima bude dostupan.

- *Settings* mapa sadrži postavke korisničkog sučelja pojedinog korisnika koje on može prilagoditi svojim željama, npr. promijeniti lozinku, izabrati vrstu editora, podesiti veličinu radnog okvira i sl.

- *Logout* je ikona kojom se pokreće odjava sa servisa, tj. završetak rada.

- *Admin* je mapa koju vide i u koju imaju pristup, samo korisnici s administratorskim ovlastima.

Unutar stablenog okvira, Tree frame-a, mogu se pokrenuti i neke operacije nad mapama. Desnim klikom miša na neku mapu pokazuje se izbornik s mogućim akcijama.

Folder actions frame

Ispod *Tree frame*-a, u donjem lijevom kutu, nalazi se Folder actions frame u kojem su prikazane trenutno dozvoljene operacije s mapama, npr. New – nova mapa, Rename – promijeni ime mape i sl. Operacija se postiže lijevim klikom na ključnu riječ.

Working frame

Najveći dio radne površine ekrana zauzima *Working frame*. Unutar *Working frame*-a odvija se rad u Scriptrunner-u. Ovisno o odabranoj akciji mijenjat će se i sadržaj *Working frame*-a.

Status frame

Status frame je statusna linija Scriptrunnera i nalazi se ispod *Working frame*-a. Na njoj se prikazuju razne obavijesti ili upozorenja o izvršenim akcijama. Statusna linija počinje sa strelicom koja mijenja boju. Ukoliko je akcija uspješno izvršena strelica poprimi zelenu boju, a ako nije uspješno izvršena, onda poprimi crvenu boju. Nakon ispisane statusne poruke o izvršenoj akciji, u zagradi je navedeno vrijeme akcije na koju se odnosi poruka.

4.1.2. Tipovi korisnika

U Scriptrunneru postoje tri vrste korisnika kojI se razlikuju po ovlastima koje imaju.

- **Običan korisnik** Scriptrunnera ima svoju *My files* mapu u kojoj može uređivati podmape i datoteke. *Public files* može pregledavati i izvoditi, a svoje sučelje uređuje preko postavki u *Settings* mapi.
- **Administrator** je korisnik koji služi za administriranje ovog servisa. On osim nabrojenih ovlasti koje ima *običan korisnik* ima i ovlasti dodavanja novih korisnika svih tipova (običnog, administratorskog i javnog), brisanja postojećih korisnika, objavljivanje prijavljenih korisničkih mapa u *Public files* mapi, te pridruživanje dozvola korisnicima na programske jezike koje oni smiju izvršavati.
- **Javni korisnik** je korisnik koji ima ovlasti samo pokretanja i pregledavanja datoteka iz *Public files* mape.

4.1.3. Tipovi mapa i datoteka

U Scriptrunneru postoje dva tipa mapa i dva tipa datoteka. Njihovo razumijevanje je vrlo bitno za ispravno korištenje Scriptrunnera.

Tipovi mapa

Prilikom stvaranja nove mape odabire se njezin tip u *Folder actions frameu*. *File mapa* sadržava samo datoteke različitih formata i tipova. Unutar nje se mogu vršiti operacije nad datotekama tj. stvaranje datoteka, brisanje, pokretanje i sl.

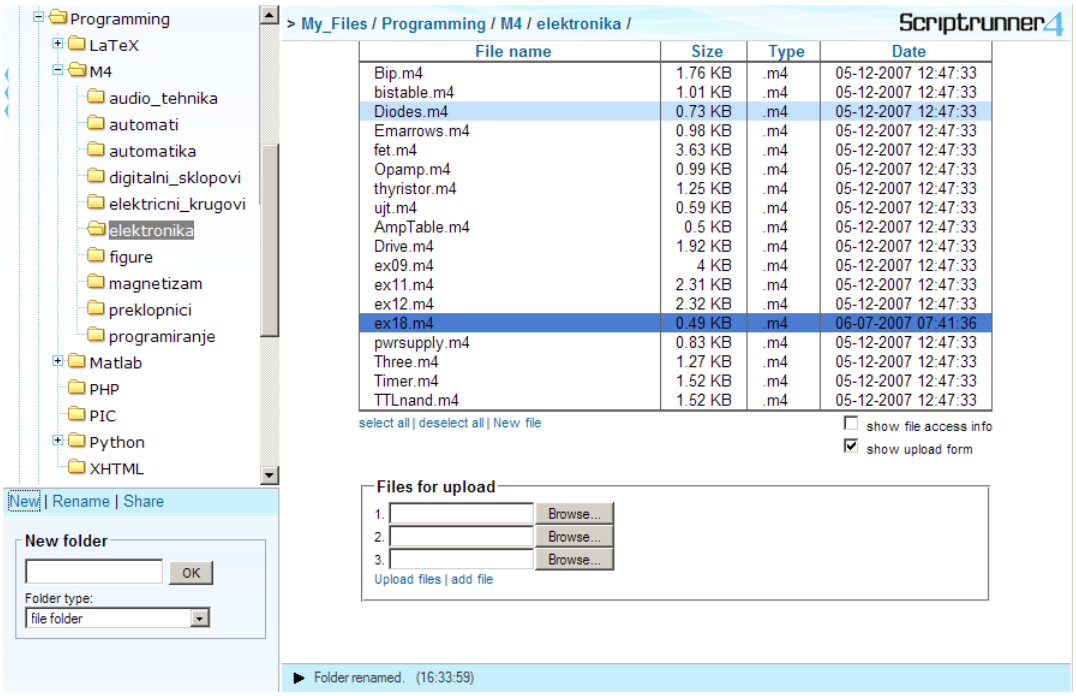
Tipovi datoteka

U Scriptrunneru postoje dva tipa datoteka:

- **Disk file** je datoteka koja se nalazi u nekoj od mapa na serveru gdje je instaliran Scriptrunner. To je obična datoteka koja može biti bilo kojeg tipa npr. slika (.jpg, .gif, .bmp), tekstualna datoteka (.txt, .doc, .pdf) ili arhiva (.zip, .rar) i za koju vrijede pravila i ograničenja operacijskog sustava na kojemu je instaliran Scriptrunner (u našem slučaju Linux-a).
- **Source file** je datoteka koja se fizički ne nalazi u nekoj od mapa na disku, nego je spremljena u MySQL bazi. Uz svaku od tako spremljenih datoteka može se spremiti opis datoteke-programa, parametri naredbene linije, vrsta programskog jezika s kojim se pokreće i sl. U ovom slučaju nema ograničenja za nazive datoteka, a mogu se spremiti i više datoteka istog imena u istoj mapi. Razlog je taj, što svaka *Source file* datoteka ima svoj jedinstveni indentifikacijski broj. To će se pokazati kao vrlo korisno svojstvo kod pisanja interaktivnih knjiga u kojima će se moći pozivati takve datoteke.

4.1.4. Rad s datotekama

Datoteke u Scriptrunneru uvijek se nalaze unutar neke mape. Korisnik može stvoriti bilo koliko mapa, a svaka mapa može sadržavati bilo koji broj datoteka. Iako postoje dvije vrste mapa, rad s datotekama je potpuno isti bez obzira na tip mape. Slika 4.3 prikazuje sadržaj jedne mape (tipa *Files*).



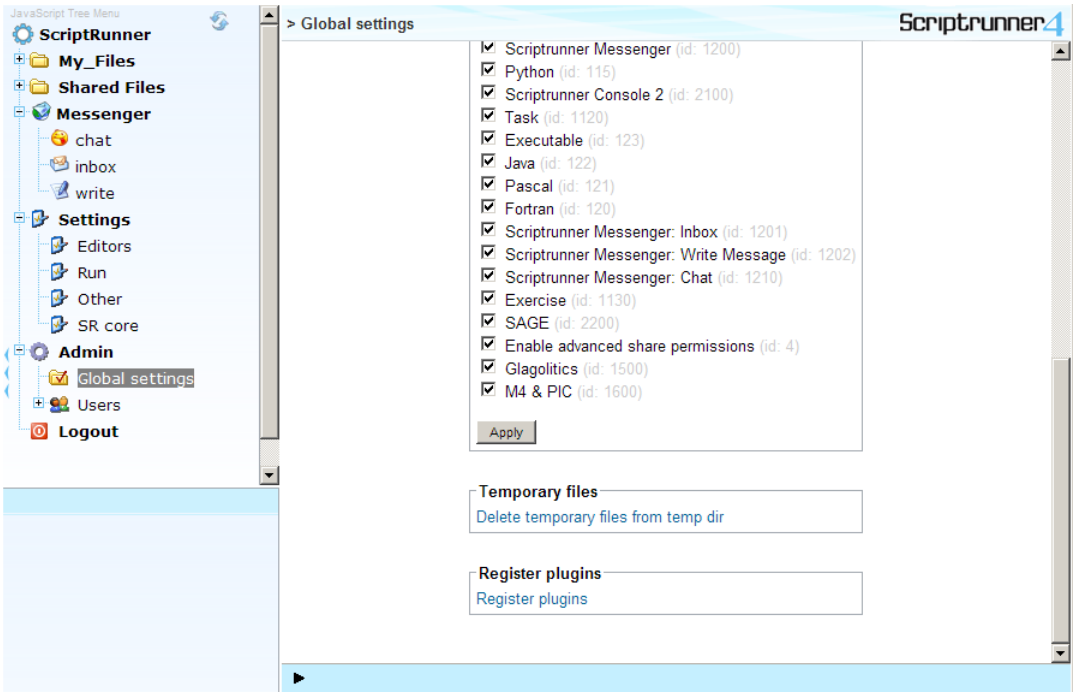
Slika 4.3: Datoteke u mapi

Lijevim klikom na bilo koju mapu dobivamo njezin sadržaj u *Working frame*-u. Na popisu se nalaze sve datoteke u odabranoj mapi. Datoteke se mogu sortirati po imenu, veličini, tipu i datumu promjene po uzlaznom ili silaznom redoslijedu, i to jednostrukim ili dvostrukim klikom na oznaku stupca tablice mape (File name, Size, Type, Date). Ispred svakog imena datoteke je *checkbox* i ukoliko je označen, na tu datoteku će se primjeniti odabrana akcija. Idući klik na isto mjesto izvrće akciju označavanja: označeno se poništava, a poništeno označava. Moguće je označiti i sve datoteke odjednom (bez da se označuju jedna po jedna), i to klikom na checkbox - select all.

Razlikuje se lijevi i desni klik miša na ime datoteke.

4.2. Scriptrunner moduli (Plug-ins)

Scriptrunner 4 omogućuje korisniku da napiše vlastiti modul koji može raditi željene operacije. Svaki modul ima svoju mapu u koju sprema svoje datoteke. Također svaki modul ima osnovnu konfiguracijsku datoteku koja se nalazi u mapi modula, a služi za povezivanje sa Scriptrunner sustavom.



Slika 4.4: Scriptrunner 4 - Moduli

Slika 4.4 prikazuje izgled administracijskog sučelje za rad s modulima. Postoji 5 tipova modula koje scriptrunner raspoznaje:

1. **folder** - modul koji je ustvari posebna web aplikacija koja se izvršava u svojem okviru (engl. *frame*), a otvara se kada korisnik klikne na direktorij u lijevom izborniku

2. **file editor** - modul za editiranje datoteka. Svi moduli ovog tipa biti će ponuđeni korisniku kroz "settings" da izabere svoj defaultni editor s kojim će uređivati datoteke

3. **run** - moduli ovog tipa služe za izvršavanje programa (npr. C, PHP, M4/PIC)

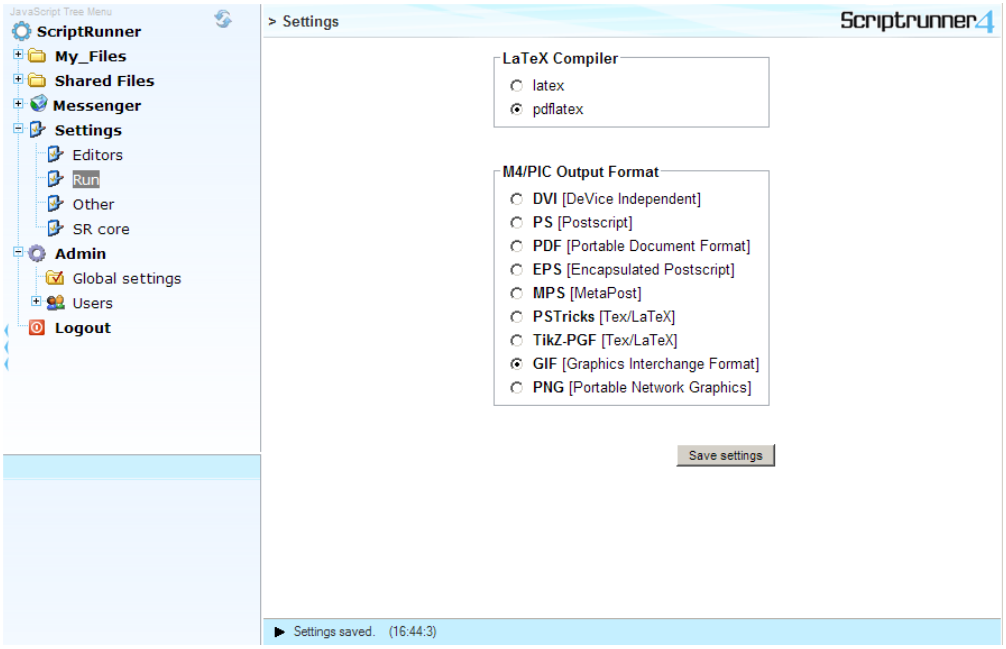
4. **file manager** - moduli ovog tipa služe za dodavanje opcije u kontekstualni izbornik file managera

5. **settings** - moduli služe za definiranje nekih postavki na razini grupa i sl. da se iskoristi postojeći sustav dozvola

U trenutku pisanja rada Scriptrunner sustav uključuje oko 40 modula.

4.3. Modul za M4/PIC preprocesor

Napravljen je M4/PIC modul, koji omogućava lako korištenje M4/PIC compilera, i korisniku nudi nekoliko grafičkih formata na izlazu, kao rezultat izvršavanja napisanog programa. Modul je napisan u PHP[1] skriptnom jeziku. Prednost ovog modula je u tome da korisniku omogući da koristi PIC jezik za crtanje shema bez ikakvih potrebnih instalacija na samo računalo korisnika. To korisniku uvelike olakšava posao, i može se orjentirati prema crtanju a ne prema instalaciji.

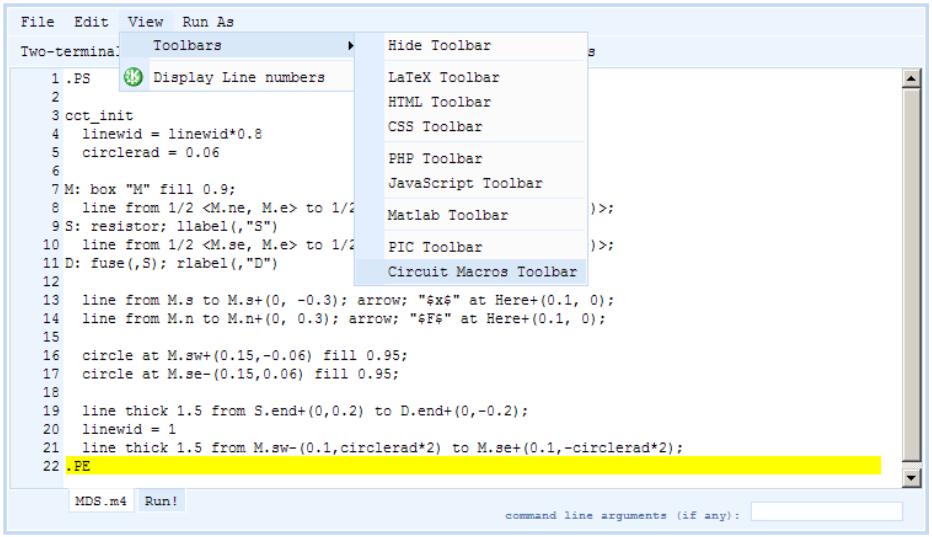


Slika 4.5: Postavke M4/PIC i LaTeX modula

Za korištenje modula potrebno je imati ovlasti koje dodjeluje administrator Scriptrunnera. Moguće je odabrati željeni format izlaza, ili u postavkama (slika 4.5), ili nakon izvršavanja modula.

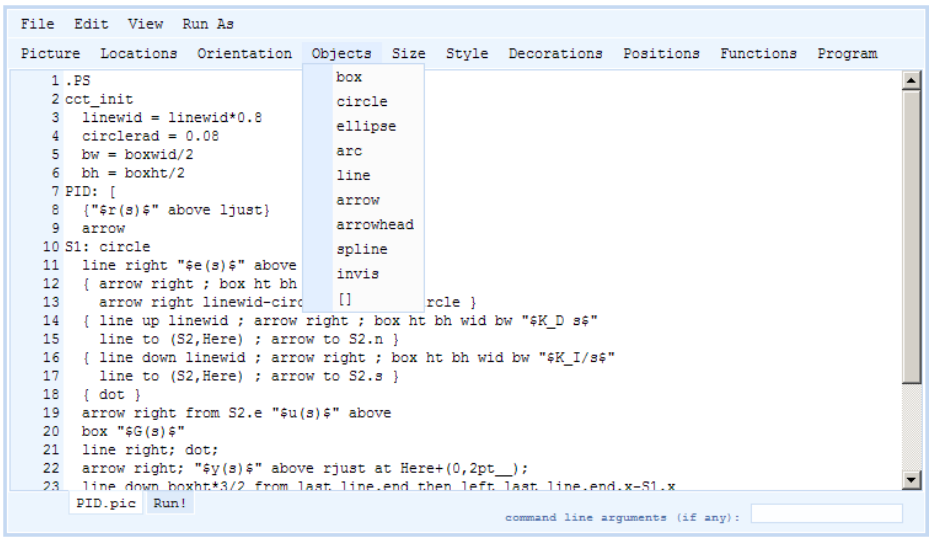
M4 ili PIC program možete pisati u jednom od dva ponuđena editora koja trenutno postoje u Scriptruner sustavu i to *SR Run Editor* i *Script Editor*. Primjer programa u *Script Editoru* prikazan je slikom 4.10. *Script Editor* raspolaže izbornike za *PIC* i *Circuit Macros* naredbe, koji korisniku olakšavaju crtanje i pronalaženje elemenata. Način odabira izbornika prikazan je slikom 4.6.

[1] http://www.php.net/



Slika 4.6: Script Editor: Odabir izbornika

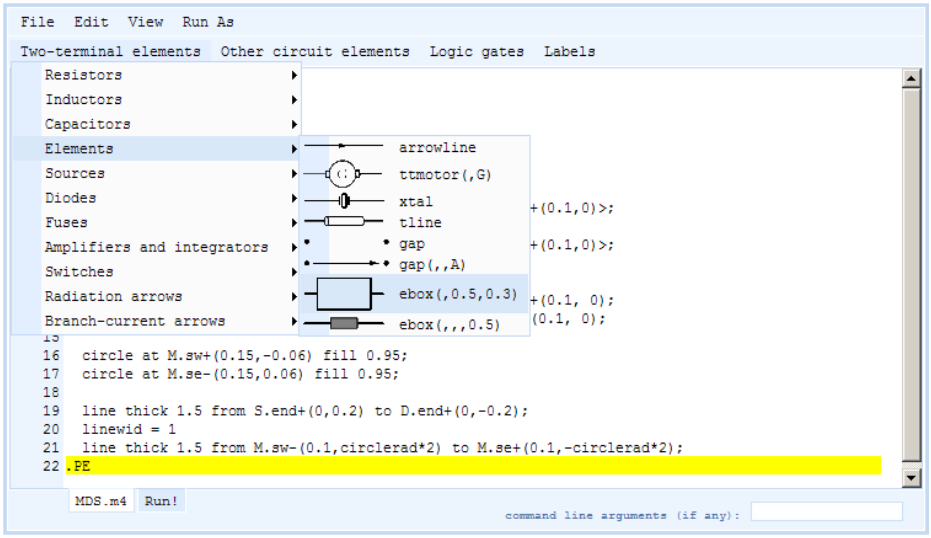
Na slici 4.7 prikazan je dio *PIC* izbornik, dok slika 4.8 prikazuje dio *Circuit Macros* izbornika.



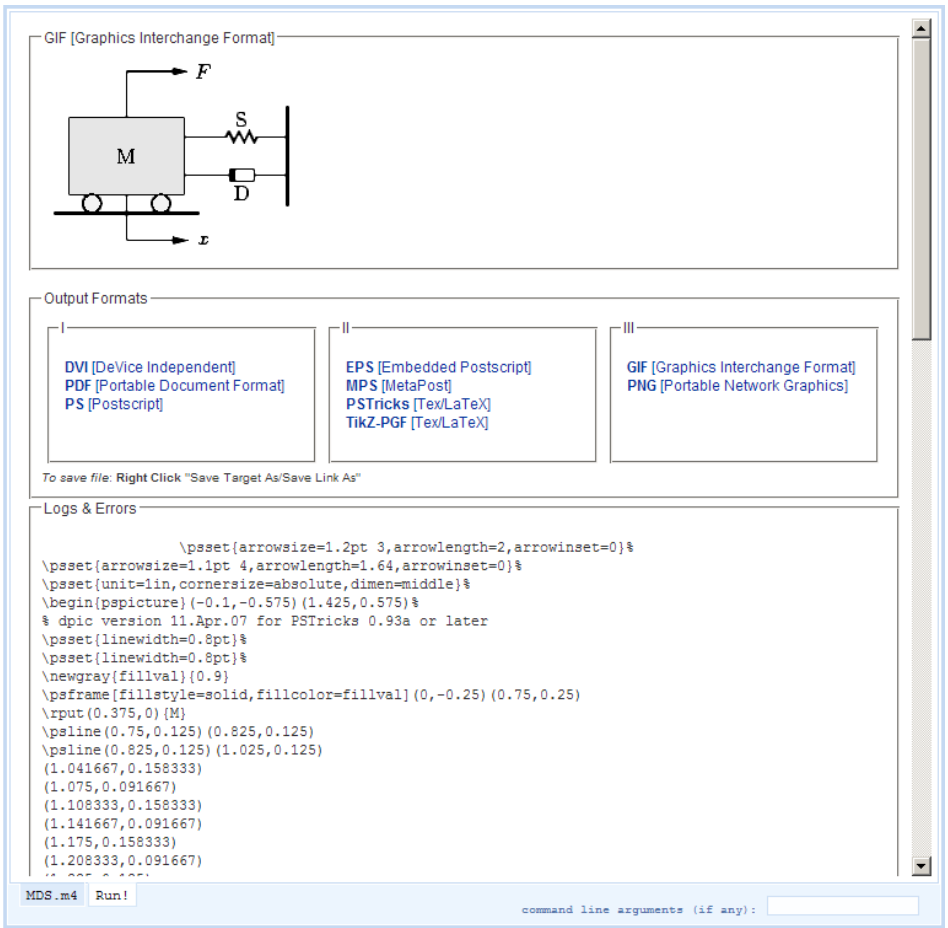
Slika 4.7: Script Editor: *PIC* izbornik

Nakon što je program napisan, potrebno ga je izvršiti pomoću M4/PIC modula. U *Script Editoru* odabere se M4/PIC modul kako je prikazano slikom 4.10.

Klikom na izbornik *Run*, pokrenut će se izvršavanje programa i prikazati shema u odabranom formatu, u našem primjeru GIF kako je prikazano slikom 4.9.



Slika 4.8: Script Editor: *Circuit Macros* izbornik

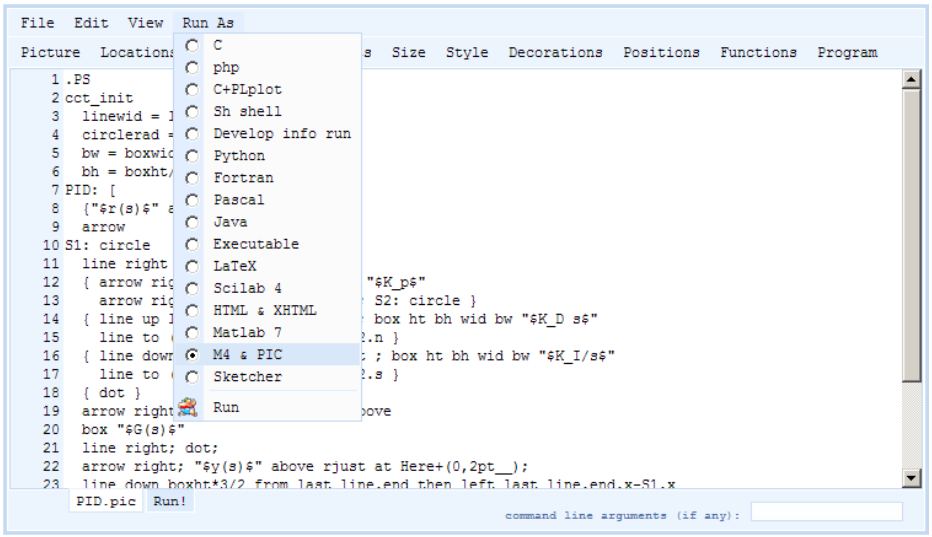


Slika 4.9: Rezultat nakon pokretanja programa: *GIF*

Shemu je također moguće dohvatiti (engl. *download*) i u drugim grafičkim formatima:

- DVI [DeVice Independent]
- PDF [Portable Document Format]
- PS [Postscript]
- EPS [Encapsulated Postscript]
- MPS [MetaPost]
- TikZ-PGF [Tex/LaTex]
- PSTricks [Tex/LaTex]
- GIF [Graphics Interchange Format]
- PNG [Portable Network Graphics]

Da bi to bilo moguće potrebno je desnom tipkom miša klinuti na željeni format, te ga pohraniti na računalo (engl. *Save As*). Na slici 4.9 također je vidljiv i *Logs Errors* dio u kojem se ispisuje obavjesti i greške koje se javljaju tijekom izvršavanja programa.

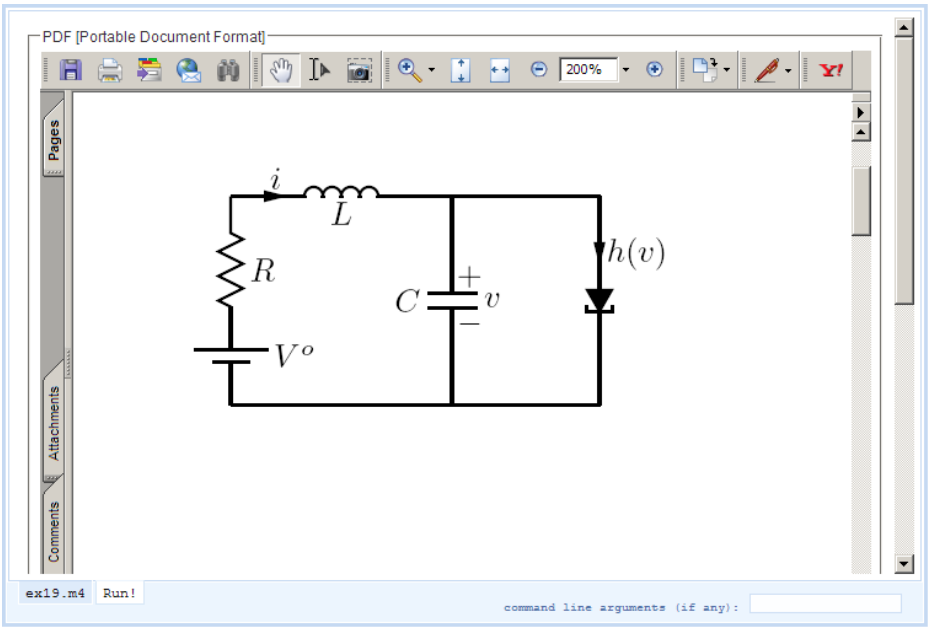


Slika 4.10: Script Editor: Odabir M4/PIC izvršnog modula

Slikom 4.11 prikazan je rezultat u PDF formatu, što je slučaj ako se u postavkama M4/PIC modula odabralo PDF za izlazni grafički format (slika 4.5).

Početna obrada programa vrši se s *m4* i *dpic* preprocesorima (moguće je koristiti i *gpic* preprocesor umjesto *dpic*). Ta obrada rezultira **mps** i **tex** (**PSTricks**, **TikZ-PGF**) datotekama. Izlazna **tex** datoteka obrađuje se sa *latex* programom. To rezultira **dvi** datotekom koja obrađena *dvips* programom daje **ps** i **eps** grafičke formate.

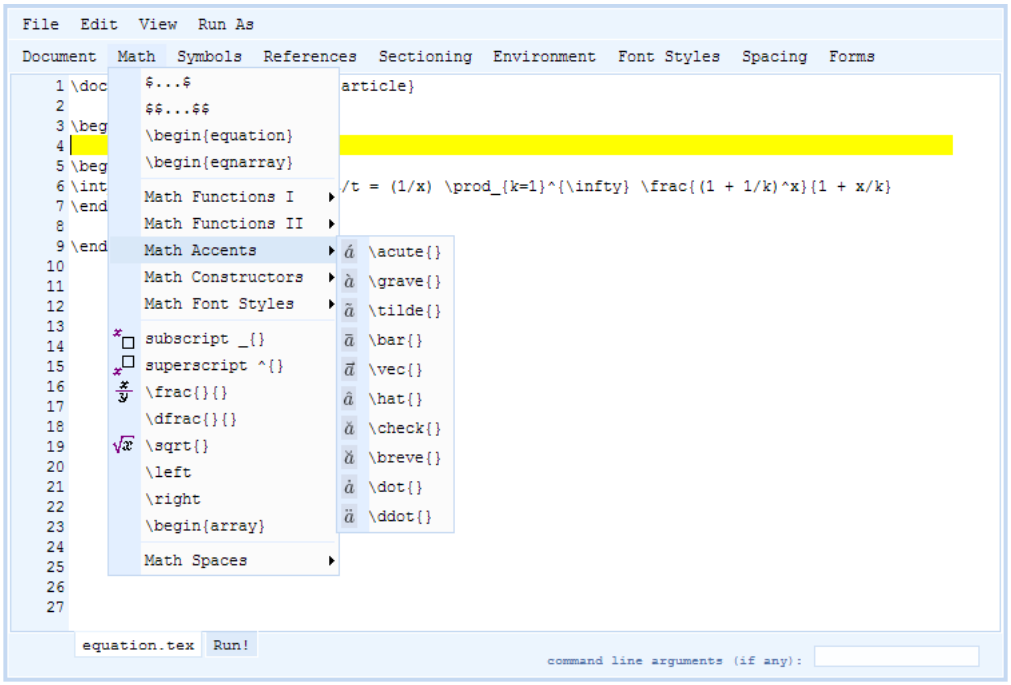
Nakon obrade **ps** datoteke s *ps2pdf* dobiva se **pdf** dokument. Iz **pdf** dokumenta pomoću programa *gs* dobivaju se **png** i **gif** grafičke datoteke.



Slika 4.11: Rezultat nakon pokretanja programa: *PDF*

4.4. Tipografski sustav LaTeX

LaTeX je programski jezik za dokumente i sustav za stvaranje dokumenata korištenjem uređivačkog program TeX. Koristi se u akademskoj zajednici kao i u izdavačkim tvrtkama, posebno zbog kvalitete krajnjeg produkta i bolje automatizacije tablica, figura i bibliografija. LaTeX je napisao Amerikanac Leslie Lamport kao zaposlenik tvrtke SRI International ranih 1980-ih i od tada je postao glavna metoda za korištenje TeX-a, jezika za formatiranje na niskoj razini.



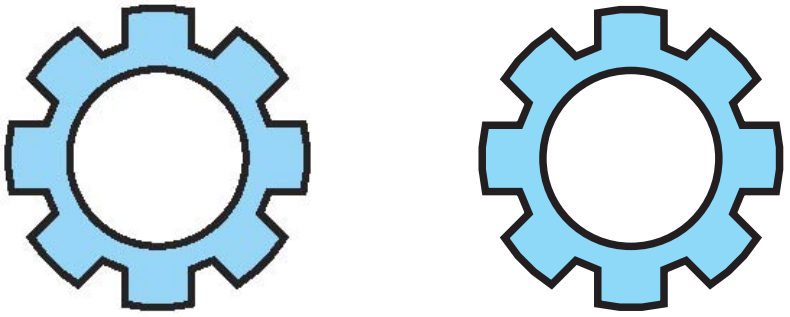
Slika 4.12: Uređivanje LaTeX dokumenta

Slika 4.12 prikazuje izgled ScriptEditora za pisanje LaTeX dokumenata unutar Scriptrunner programa. Također za lakše pisanje postoji padajući izbornik sa raznim elementima. Tako napisani dokument potrebno je izvršiti s LaTeX modulom koji u postavkama dozvoljava mijenjanje programa kojim se modul izvršava kako je prikazano na slici 4.5.

Scriptruner ima ugrađene biblioteke za izradu interaktivnih sadržaja (eng. *forms*) u LaTeX modulu koje su korištene u diplomskom radu. To koriskniku omogućuje lakše učenje i čitanje.

4.5. Grafički formati

Danas su zastupljena dva grafička formata i to *rasterska grafika* i *vektorska grafika*. Svaki od ova dva formata koristi se za određene tipove slika. Tako je u rasterskom formatu bolje spremiti fotografiju, a u vektorskom tehničku shemu. Slika 4.13 prikazuje logo Scriptrunner programa zapisanog u rasterskoj i vektorskoj grafici.



Slika 4.13: Primjer rasterske i vektorske grafike

4.5.1. Rasterska grafika

Rasterska grafika je pojam koji opisuje sliku sastavljenu od sićušnih obojenih kvadratića, zvanih pikseli. Osnovna osobina ove grafike je da ima određenu rezoluciju i ne može se uvaćavati i smanjivati bez gubitka kvalitete. Odnosno točkice od kojih je slika stvorena postaju vidljive. Kod takvih programa, kada povučete liniju on je pretvoti u sićušne kvadratiće poslagane jedan do drugog koji tvore jednostavnu sliku. To se postiže programima kao što su: Photo-Paint, Micrografxov Picture Publisher i Uleadov PhotoImpact.

Najčešći rasterski formati grafičkih datoteka:

- .**BMP** (Windows Bitmap)
  Koristi se za prikaz i spremanje Windows slika.
- .**DCX** (Windows Images)
  Format za višestruke '.PCX' datoteke.
- .**GIF** (Graphics Interchange Format)
  Ovaj format može se koristiti na različitim platformama, pa se često koristi za spremanje slika za WWW. Pošto je zapis relativno mali, slike se brzo prenose putem Interneta. No, GIF je ograničen na 256 boja, i boje nisu prikazane istovjetno u ovisnosti o platformama.
- .**JPG** (Joint Photographic Experts Group)
  Internacionalni standard korišten za kompresiju digitalnih slika. Datoteke su relativno male, ovisno o stupnju kompresije. To je format kojim se ne može povratiti orginalna slika.

- .**PCX** (PC Paintbrush)
  Format kreiran od Zsoft. Podržan od mnogih Windows aplikacija, kao i od mnogih optičkih skenera i fax modema.

- .**PNG** (Portable Network Graphics)
  Otvoreni format i koristi kompresiju bez gubitaka. Ovaj format u zadnje vrijeme postaje sve popularniji, a nastao je kao alternativa svepopularnom GIF formatu. Podržavaju ga svi noviji browseri, ima mogućnost prozirne pozadine. PNG je zamišljen kao grafički format za razmijenu preko interneta, a ne za profesionalnu uporabu.

- .**TIF** (Tagged Image File Format)
  Ovaj format se koristi za spremanje i razmjenu između 'desktop publishing' i 'graphic design' aplikacija. TIFF također podržavaju različite platforme, poput Microsoft Windows-a i Macintosh-a. Također se koristi za skeniranje slika jer podržava sve veličine, rezolucije i dubine boja.

4.5.2. Vektorska grafika

Vektorska grafika je definirana funkcijama koje određuju crte, oblike i položaje. Sastoji se isključivo od objekta u slici. Objekti mogu biti linije, krugovi, kvadrati, trokuti, zavojnice ili drugi oblici. Vektorske slike obično su sastavljene od linija. Umjesto da sliku crtamo točku po točku, kao u paint programima, u vektorskoj grafici sliku crtamo matematičkim funkcijama. Vektorsku se grafiku može beskonačno povećavati i smanjivati. Bez obzira na to koliko velikom učinite svoju sliku, uvijek izgleda savršeno. Zapravo grafika može izgledati i bolje kada je uvećana zato što su krivulje glađe. Programi za stvaranje vektorskih grafika su AutoCad, Corel Draw, i sl.

Najčešći vektorski formati grafičkih datoteka:

- .**CGM** (Computer Graphics Metafile)
  Format koji je razvijen suradnjom različitih organizacija za standardizaciju. Podržan je od mnogih softverskih produkata.

- .**DXF** (Data Exchange File)
  Format kreiran od AutoDesk-a. Skoro svi PC bazirani CAD sustavi podržavaju

DXF.

- .**EPS** (Encapsulated Postscript)
  Format za PostScript jezik. EPS koristi kombibaciju PostScript komandi i TIFF ili PICT format.

- .**MPS** (MetaPost)
  Format kreiran od John Hobby, a temelji se na Donald Knuthovom metafont, ali je dodana podrška za izlaz u PostScript.

- .**PIC** (Lotus Picture File)
  Relativno jednostavan grafički format razvijen od Lotus-a za prikaz grafike generirane Lotusom 1-2-3. PIC je podržan od mnogih PC aplikacija.

- .**PICT** (Picture Format)
  Format za Macintosh grafičke datoteke razvijen od Apple Computer. On je podržan od svih grafičkih programa koji rade na Macintosh-u.

- .**WMF** (Windows Metafile)
  Format za spremanje i razmjenu slika za Windows aplikacije.

- .**WPG** (WordPerfect Graphic File)
  Format koji koristi WordPerfect.

5 Grafički sustavi PGF i Tikz

5.1. Proširenje grafičkih mogućnosti u LaTeX-u

Korisnici TeX i LaTeX programa imaju višestruke mogućnosti stvaranja i upotrebe grafike u svojim dokumentima. Postoji desetak sustava koji to omogućuju, poput PIC sustava koji je obrađen u drugom poglavlju. Cilj svih takvih sustava je stvaranje grafike koja se jednostavno uključuje unutar LaTeX naredbi i ne zahtijeva vanjski program za crtanje. Štoviše, današnji zahtjevi temelje se na automatskom generiranju ili uključivanju vektorizirane grafike, tako da povećanje ili promjena dokumenta ne utječe na izgled grafičkih objekata.

Jedan od modernih sustava (zadnja verzija 1.01, 2005. godine) je PGF/TikZ grafički sustav, koji zadovoljava navedene uvjete i kao rezultat daje PDF, PostScript i SVG. Ovaj sustav djeluje u paru: **PGF** nudi niz grafičkih funkcija niske razine (engl. *low-level graphics primitives*), dok **TikZ** daje korisničko sučelje visoke razine. Slično kao u macro naredbama elektroničkih sustava i ovdje se u *TikZ*-u pišu naredbe visoke razine, koje se onda pretvaraju u niz *PGF* upravljačkih naredbi koje se izvršavaju (poput PIC naredbi). TikZ naredbe pišu se na isti način kao LaTeX naredbe, štoviše, TikZ je poseban (*tikzpicture*) okoliš (engl. *environment*), sličan onom grafičkom ili matematičkom u LaTeX-u.

U najednostavnijem slučaju radi se o naredbama koje povezuju točke u ravnini ili upotreba složenijih objekata kao što su pravokutnici, kružnice, lukovi, tekst, mreže, Bézier-ove krivulje i sl. Posebna značajka takvog pristupa je nesmetana upotreba TeX fontova, simbola i matematike unutar tako generirane grafike.

Sljedeći primjer prikazuje izgled LaTeX dokumenta temeljenog na TikZ-u.

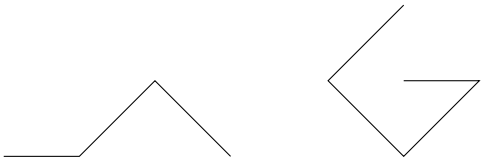
```
\documentclass[11pt]{article}
...
\usepackage{tikz}
% PGF biblioteke
\usepackage{pgflibraryarrows}
\usepackage{pgflibrarysnakes}
...
\begin{document}
...
\begin{tikzpicture}
...
\end{tikzpicture}
...
\end{document}
```

Osnova ideja koja se koristi u TikZ-u su točke i staze. Točke se mogu definirati na četiri načina:

- u apsolutnim kartezijevim koordinatama(x,y), gdje su x i y izražene u centimetrima
- u polarnim koordinatama (modul, kut), gdje se dimenzije zadaju u centimetrima, pointima ili sl.
- preko imenovanih točki, npr. *\path (a,b) coordinate (P);*
- preko relativnih pomaka, npr. *\path (P) ++(dx,dy) coordinate (Q);*

Sintaksa pisanja naredbi (staza) od jedne točke do druge slična je MetaPost grafičkom pristupu. Primjer prikazan slikom 5.1, povezuje relativne točke predočen je sljedećim programom:

```
\begin{tikzpicture}
    % iduća točka mijenja se s obzirom na prethodnu
    \draw (0,0) -- ++(1,0) -- ++(1,1) -- ++(1,-1);
    % iduća točka mijenja se s obzirom na početnu
    \draw (0,0) -- +(1,0) -- +(0,-1) -- +(-1,0) -- +(0,1);
\end{tikzpicture}
```

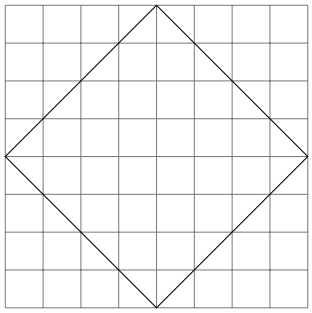


Slika 5.1: Relativne točke

Unutar TikZ naredbe znakovi '–' povlače liniju između početne i krajnje točke. Kao u C-jeziku ++() znači prethodno uvećavanje (obaju) koordinata, dok jednostruki +() označuje pribrajanje novih koordinatnih vrijednosti početnoj točki.

Za potrebe traženja pogreški (engl. *debugging*) korisno je na početku crtanja uključiti mrežu (engl. *grid*) kao u sljedećem primjeru (slika 5.2):

```
\begin{tikzpicture}
   \draw[step=0.25cm,color=gray] (-1,-1) grid (1,1);
   \draw (1,0) -- (0,1) -- (-1,0) -- (0,-1) -- cycle;
\end{tikzpicture}
```



Slika 5.2: Apsolutne točke s mrežom

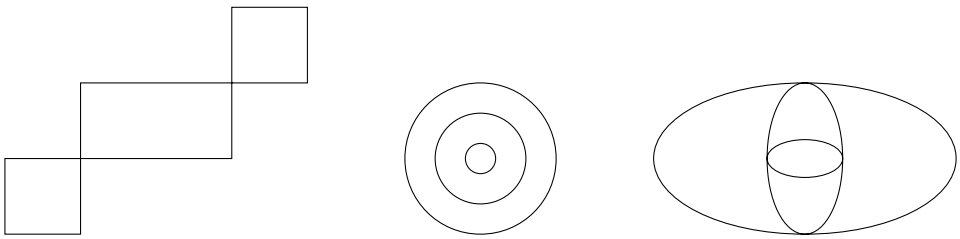
Mreža je određena dvjema krajnjim točkama *(-1,-1)* i *(1,1)*, razmakom između nutarnjih linija (*step=0.25cm*) i bojom (*color=gray*).

5.2. Složeniji grafički oblici

Složeniji grafički oblici nastaju od jednostavnijih, a obuhvaćeni su prikladnim TikZ naredbama. Tako će pravokutnik (*rectangle*) biti mreža bez nutarnjih linija (s definicijom krajnjih točaka jednako kao mreža), a zakrivljene linije tvorit će lukove (*arc*), kružnice (*circle*), elipse (*ellipse*) i sl. Nakon crtanja složenog lika,

točka nastavka je u zadnje definiranoj točki. Tako se s jednom naredbom može nacrtati više objekata. Sljedeći primjer prikazan je slikom 5.3.

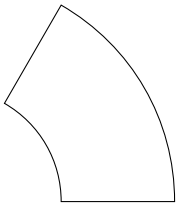
```
\begin{tikzpicture}
    \draw (0,0) rectangle (1,1)
        rectangle (3,2)
        rectangle (4,3);

    \draw (0,0) circle (1cm)
        circle (0.6cm)
        circle (0.2cm);

    \draw (0,0) ellipse (2cm and 1cm)
         ellipse (0.5cm and 1 cm)
         ellipse (0.5cm and 0.25cm);
\end{tikzpicture}
```



Slika 5.3: Složeniji grafički oblici

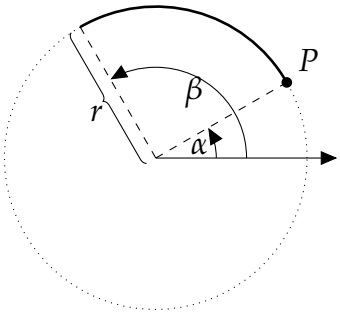
Treba primjetiti kako kod složenijeg grafičkog objekta (središte kod kružnice ili elipse, a krajnja točka kod pravokutnika) točka nastavka ostaje na zadnje definiranoj koordinati, bez obzira koliko se istih (ili sličnih) objekata na njega nadovezuje. Dakako, moguća je i kombinacija jednostavnih i složenijih oblika (slika 5.4):

```
\begin{tikzpicture}
    \draw (0:1cm) -- (0:2cm)
        arc (0:60:2cm) -- (60:1cm)
        arc (60:0:1cm) -- cycle;
\end{tikzpicture}
```



Slika 5.4: Kombinacija jednostavnih i složenijih grafičkih oblika

Osnova naredba za crtanje luka je *\draw (P) arc (alfa:beta:dim);* s parametrima kao na slici 5.5 i parametrom *dim* koji označuje dimenziju u nekim jedinicama, npr. cm ili pt.



Slika 5.5: Crtanje luka u TikZ programu

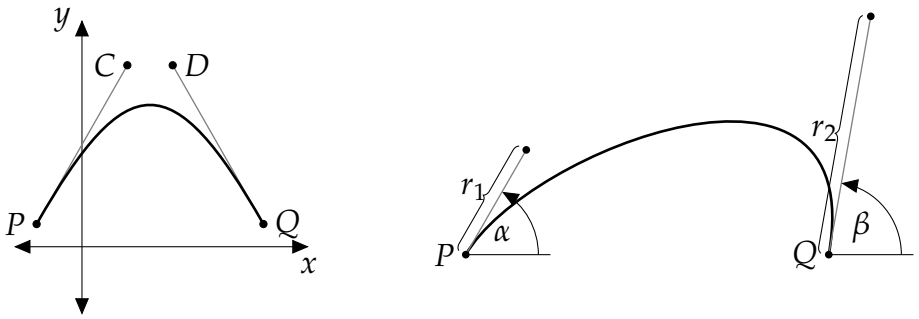
Uključivanjem točaka upravljanja (*controls*) moguće je crtati Bézier-ove krivulje, kod kojih postoji početna (P) i završna (Q) točka, a dodatne dvije točke (C i D) služe za upravljanje nagibom i smjerom krivulje. Sljedeće naredbe nacrtat će krivulje na slici 5.6:

```
\draw (P) .. controls (C) and (D) .. (Q);
\draw (P) .. controls +(a:r1:dim) and +(b:r2:dim) .. (Q);
```

Čvor (engl. *node*) je generalizirana koordinata, koja je karakterizirana oblikom i tekstom. Tako će:

```
\path (0,0) node[draw,shape=circle] (v0) {$v_0$};
```

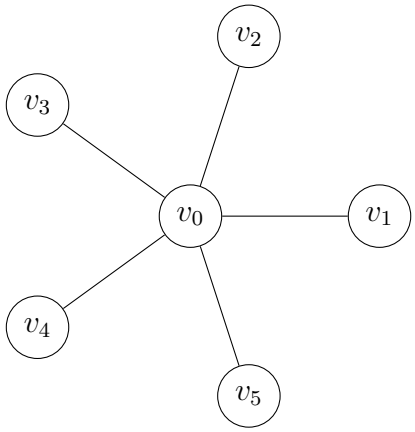
definirati čvor s imenom *v0* koji je centriran u ishodištu, s oblikom kruga i tekstom koji se piše kao matematički unutar *$$* znakova. *Path* naredba pridružuje točki



Slika 5.6: Bézier-ove krivulje

koordinatu zapisanu u kartezijevim ili (u ovom slučaju) polarnim koordinatama (*kut : modul*). Povezivanje više čvorova u graf dobije se na jednostavan način, kako pokazuje sljedeći programski odsječak i pripadna slika 5.7:

```
\begin{tikzpicture}[scale=2.5]
    \tikzstyle{every node}=[draw,shape=circle];
    \path (0:0cm)
        node (v0) {$v_0$};
    \path (0:1cm)
        node (v1) {$v_1$};
    \path (72:1cm)
        node (v2) {$v_2$};
    \path (2*72:1cm)
        node (v3) {$v_3$};
    \path (3*72:1cm)
        node (v4) {$v_4$};
    \path (4*72:1cm)
        node (v5) {$v_5$};
    \draw (v0) -- (v1)
        (v0) -- (v2)
        (v0) -- (v3)
        (v0) -- (v4)
        (v0) -- (v5);
\end{tikzpicture}
```



Slika 5.7: Čvorovi u TikZ-u

5.3. Programiranje u TikZ-u

TikZ omogućuje izvođenje programskih petlji koje stvaraju stanovite grafičke tipove. Osnovna sintaksa za programsku petlju je:

```
\foreach \var in {iteracijska lista}
{
    tijelo petlje
}
```

Varijabla $\backslash var$ uzima vrijednosti iz iteracijske liste. Na primjer, programski odsječak:

```
\foreach \i in {1,...,4}
{
    \path (\i,0) coordinate (X\i);
    \fill (X\i) circle (3pt);
}
```

načinit će četiri koordinate X1 do X4 s vrijednostima (1,0), (2,0), (3,0) i (4,0). Na koncu će se na svakoj koordinati nacrtati mali zasjenjeni kružić (polumjera 3 pt). Iteracijska lista ne mora se sastojati od sljednih cijelih brojeva. Implicitna veličina koraka dobije se zadavanjem prvih dviju vrijednosti i konačnom vrijednosti u listi. Programski odsječak:

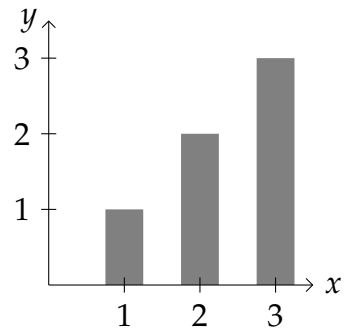
```
\foreach \angle in {0,60,...,300}
{
    tijelo petlje
}
```

pridjelit će varijabli *\angle* vrijednosti u oblika 60k gdje je $0 \leq k \leq 5$. Na sličan način, specifikacija parova vrijednosti u iteracijskoj listi daje istodobne iteracije nad ovim vrijednostima. Tako će odsječak:

```
\foreach \angle / \c in {0/red,120/green,240/blue}
{
    Tijelo petlje
}
```

načiniti tri iteracije po tijelu petlje, pridružujući parove (*0,red*), (*120,green*) i (*240,blue*) varijablama *\angle* i *\c*. Na iterativan način crtaju se i točke iz zadane liste koja može biti načinjena na različite načine (iz nekog programa s formulom, datoteke ili liste):

```
\draw plot function{gnuplot formula};
\draw plot file{filename};
\draw plot coordinates{point sequence};
```



Slika 5.8: Grafikon u TikZ-u

Tako će odsječak:

```
\draw[ycomb,color=gray,line width=0.5cm]
    plot coordinates{(1,1) (2,2) (3,3)};
```

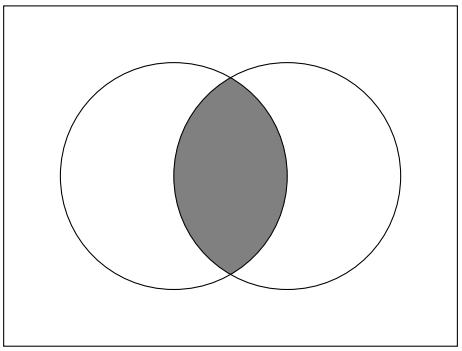
nacrtati grafikon, koji je prikazan na slici 5.8:

Kao posljednja (iako ne jedina) značajka TikZ programa je naredba *clip*, kojom se definira područje na kojem će slika biti nacrtana. Sve izvan tog područja neće biti prikazano. Pritom postoji i mehanizam dosega (engl. *scoping*) koji povezuje definirano '*clipping*' područje s nizom naredbi koje na njemu mogu djelovati.

Kao primjer neka posluži sljedeći programski odsječak i pripadna slika 5.9:

```
\begin{tikzpicture}
    \draw (-2,1.5) rectangle (2,-1.5);
    \begin{scope}
        \clip (-0.5,0) circle (1cm);
        \clip (0.5,0) circle (1cm);
        \fill[color=gray] (-2,1.5)
        rectangle (2,-1.5);
    \end{scope}
    \draw (-0.5,0) circle (1cm);
    \draw (0.5,0) circle (1cm);
\end{tikzpicture}
```

koji označuje sivom bojom samo područje u presjeku dvaju krugova.



Slika 5.9: Scope i \clip akcija

5.4. Uključivanje tehničkih slika u LaTeX

Za pisanje bilo kakvog dokumenta LaTeX programu, vrlo je vjerojatno da ćemo trebati u njega uključiti nekoliko slika. Ovisno o tome da li za izvršavanje našeg dokumenta koristimo LaTeX ili pdfLaTeX program razlikuju se grafički formati koje možemo uključiti u LaTeX dokument.

Slike se uključuju koristeći *includegraphics naredbu*, koja ima sljedeću sintaksu:

```
\includegraphics[opcije]{datoteka}
```

gdje su opcije prikazane tablicama 5.1, 5.2 i 5.3. Kako naredba *includegraphics* ne završava odjeljak, moguće je sliku uključiti unutar teksta.

Korisnik treba navesti koji grafički driver želi koristiti, između 18 ponuđenih. Preporučuje se korištenje jednog od dva najkorištenija drivera: *dvips* za DVIPS tipove datoteka i *pdftex* za pdfLaTeX datoteke. Ukoliko se korisnik odluči za korištenje jedan od ta dva drivera, ne treba to posebno navoditi jer datoteka *graphics.cfg* u većini LaTeX distribucija sama o tome vodi računa. Grafički driver poziva pomoću:

```
\usepackage{graphics}
```

Tablica 5.1: *includegraphics* opcije

Opcija	**Opis**
height	visina slike
totalheight	ukupna visina slike
width	širina slike
scale	veličina slike, npr: *scale=2* povečat će sliku dva puta
angle	kut rotacije u stupnjevima, u smjeru suprotnom od kazaljke na satu
origin	os rotacije
bb	označava BoundingBox parametre

Tablica 5.2: *includegraphics* cropping opcije

Opcija	**Opis**
viewport	označava koji dio slike da se prikaže
trim	alternativni parametar za prikazivanje dijela slike

Tablica 5.3: *includegraphics* bolean opcije

Opcija	**Opis**
clip	*clip=false* prikazuje se cijela slika bez obzira na cropping opcije, u protivnom *clip=true*
draft	*draft=true* spriječava uključivanje slike u dokument, prikazuje se okvir i ime datoteke, u protivnom prikazuje se cijela slika
keepaspectratio	zadržava proporcije između visine i širine slike

5.4.1. Uključivanje grafike za DVIPS tipove datoteka

Najbolji podržani grafički format za *dvips* tipove datoteka je *EPS*. Prilikom izvršavanja LaTeX dokumenta, sljedeća naredba

```
\includegraphics{shema.eps}
```

uključuje sliku iz *eps* datoteke *shema.eps* u prirodnoj veličini. Ukoliko bi se izvršila sljedeća naredba

```
\includegraphics{shema}
```

tada bi *includegraphics* naredba pokušala pridjeliti dodatak (eng. *extension*) iz *DeclareGraphicsExtensions* popisa dodataka.

5.4.2. Uključivanje grafike za pdfLaTeX datoteke

pdfLaTeX podržava direktno uključivanje *pdf*, *png*, *jpeg*, i *MetaPost* grafičkih formata. Prilikom izvršavanja dokumenta s pdfLaTeX sljedeće naredbe:

```
\includegraphics{shema.pdf}
\includegraphics{shema.png}
\includegraphics{shema.jpg}
\includegraphics{shema.mps}
```

uključuju slike iz *pdf* datoteke *shema.pdf*, *png* datoteke *shema.png*, *jpg* datoteke *shema.jpg* i *mps* datoteke *shema.mps* u njihovoj prirodnoj veličini. Ukoliko bi se izvršila sljedeća naredba

```
\includegraphics{shema}
```

tada bi *includegraphics* naredba pokušala pridjeliti dodatak iz *DeclareGraphicsExtensions* popisa dodataka.

5.4.3. *DeclareGraphicsExtensions* naredba

Naredba *DeclareGraphicsExtensions* govori LaTeX-u koji dodatak da koristi u slučaju da on nije definiran kod pozivanja *includegraphics* naredbe. Ovisno o korištenom grafičkom driveru, razlikuju se dodaci koji se provjeravaju. Npr. u slučaju ako se koristi *dvips* grafički driver, sljedeći dodaci se provjeravaju

```
\DeclareGraphicsExtensions{.eps,.ps,.eps.gz,.ps.gz,.eps.Z}
```

Koristi li se pak *pdftex* grafički driver, provjeravaju se sljedeći dodaci:

```
\DeclareGraphicsExtensions{.png,.pdf,.jpg,.mps}
```

Ukoliko bi se izvršila sljedeća naredba, pri korištenju *dvips* grafičkog drivera,

```
\includegraphics{shema}
```

LaTeX izvršitelj bi prvo tražio datoteku *shema.eps*, pa *shema.ps*, pa *shema.eps.gz* dok se ne pronađe postojeća datoteka, što nam omogućava da kod pozivanja datoteke ne trebamo upisati dodatak.

U slučaju da koristimo oba LaTeX i pdfLaTeX izvršitelja, sljedećim naredbama

```
\usepackage{ifpdf}
...
\ifpdf
```

```
    \DeclareGraphicsExtensions{.pdf,.png,.jpg,.mps}
\else
    \DeclareGraphicsExtensions{.eps}
\fi
```

moguće je definirati željeni redoslijed traženja datoteka.

5.4.4. Pozicioniranje slike unutar LaTeX dokumenta

Želimo li uključiti tehnički shemu *shema.eps* unutar LaTeX dokumenta, to možemo učiniti sljedećim naredbama:

```
\begin{figure}
    \centering
    \includegraphics[totalheight=5sm]{shema.eps}
    \caption{Opis sheme}
    \label{shema1}
\end{figure}
```

Tako uključena shema pozicionira se na mjestu gdje su naredbe napisane. Naredba *label* numerira sliku, te se pomoću nje može pozvati na sliku pomoću *ref* ili *pageref* naredbe. *label* naredba mora se pozvati odmah nakon *caption* naredbe u kojoj se navodi opis slike.

Figure okružje ima dodatne opcije za pozicioniranje slike unutar dokumenta, što korisniku omogućuje da odredi željenu poziciju slike. Dodatne opcije mogu koristiti bilo koju kombinaciju sljedećih slova:

- **h** (Here): Postavlja sliku u tekstu gdje je naredba pozvana. Ova opcija se nemože pozvati ako nema dovoljno mjesta na stranici za sliku.
- **t** (Top): Postavlja sliku na vrhu stranice.
- **b** (Bottom): Postavlja sliku na dnu stranice.
- **p** (Float Page): Postavlja sliku da slobodno stoji na stranici

U slučaju da nisu zadane opcije za pozicioniranje one se automatski postavljaju na [tbp]. Najbolje je koristiti jednu od sljedećih opcija: [htbp], [tbp], [htp] ili

[tp], dok korištenje samo jedne opcije [t], [b], [p] ili [h], najčešće izazvia problem s pozicioniranjem.

PSTricks

5.5. PSTricks biblioteke za električne elemente

Postoje mnoge *PSTricks* biblioteke koje služe za crtanje raznih električnih elemenata, blokova diagrama, ili nekih drugih elemenata. Jedna od njih je *pst-circ*, a služi za crtanje električnih elemenata i shema. Autori ove biblioteke su Christophe Jorssen i Herbert Voß†. Zadnja verzija objavljena je 15. rujna 2007. godine.

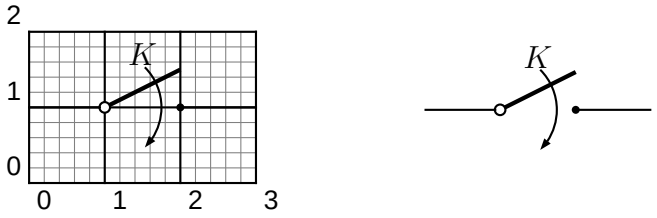
Da bi se mogla koristiti *pst-circ* biblioteka potrebno je u zaglavlju LaTeX dokumenta uključiti sljedeće naredbe:

```
\usepackage{pstricks}
\usepackage{pst-circ}
```

Biblioteka sadrži 50-ak macro naredbi kojima se crtaju elementi. Slijedi nekoliko primjera macro elemenata i shema nacrtanih pomoću ove LaTeX biblioteke.

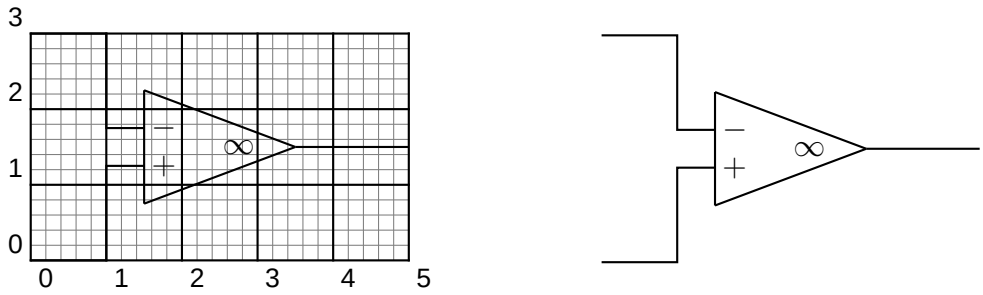
Primjer sklopke opisan je sljedećim programom, a prikazuje ga slika 5.10. Želi li se nacrtati i mreža, potrebno je uključiti naredbu *psgrid*.

```
\begin{pspicture}(3,2)\psgrid
    \pnode(0,1){A}
    \pnode(3,1){B}
    \switch(A)(B){$K$}
\end{pspicture}
```



Slika 5.10: Električki element: sklopka

```
\begin{pspicture}(5,3)\psgrid
    \pnode(0,0){A}
    \pnode(0,3){B}
    \pnode(5,1.5){C}
    \OA(B)(A)(C)
\end{pspicture}
```



Slika 5.11: Električki element: sklopka

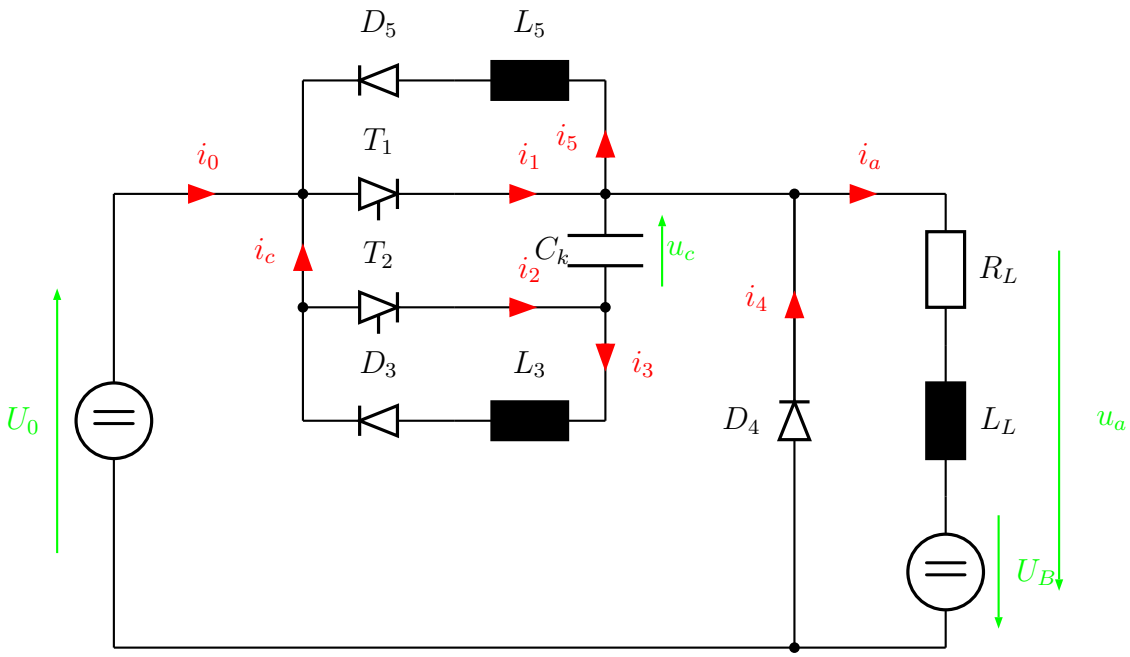
Sljedeći program nacrtati će električnu shemu prikazanu slikom 5.12.

```
\begin{pspicture}(0,-0.2)(13,8)
    \psset{intensitycolor=red,intensitylabelcolor=red,
        tensioncolor=green,tensionlabelcolor=green,
        intensitywidth=3pt}
    \circledipole[tension,tensionlabel=$U_0$,
        tensionoffset=0.75,
        labeloffset=0](0,0)(0,6){\LARGE\textbf{=}}
    \wire[intensity,intensitylabel=$i_0$](0,6)(2.5,6)
    \diode[dipolestyle=thyristor](2.5,6)(4.5,6){$T_1$}
    \wire[intensity,intensitylabel=$i_1$](4.5,6)(6.5,6)
    \multidipole(6.5,7.5)(2.5,7.5)%
    \coil[dipolestyle=rectangle,labeloffset=-0.75]{$L_5$}%
    \diode[labeloffset=-0.75]{$D_5$}.
    \wire[intensity,intensitylabel=$i_5$](6.5,6)(6.5,7.5)
    \wire(2.5,7.5)(2.5,3)
```

```
    \wire[intensity,intensitylabel=$i_c$](2.5,4.5)(2.5,6)
    \qdisk(2.5,6){2pt}\qdisk(6.5,6){2pt}
    \diode[dipolestyle=thyristor](2.5,4.5)(4.5,4.5){$T_2$}
    \wire[intensity,intensitylabel=$i_2$](4.5,4.5)(6.5,4.5)
    \capacitor[tension,tensionlabel=$u_c$,tensionoffset=-0.75,
        tensionlabeloffset=-1](6.5,4.5)(6.5,6){$C_k$}
    \qdisk(2.5,4.5){2pt}\qdisk(6.5,4.5){2pt}
    \wire[intensity,intensitylabel=$i_3$](6.5,4.5)(6.5,3)
    \multidipole(6.5,3)(2.5,3)%
    \coil[dipolestyle=rectangle,labeloffset=-0.75]{$L_3$}%
    \diode[labeloffset=-0.75]{$D_3$}.
    \wire(6.5,6)(9,6)\qdisk(9,6){2pt}
    \diode(9,0)(9,6){$D_4$}
    \wire[intensity,intensitylabel=$i_4$](9,3.25)(9,6)
    \wire[intensity,intensitylabel=$i_a$](9,6)(11,6)
    \multidipole(11,6)(11,0)%
    \resistor{$R_L$}
    \coil[dipolestyle=rectangle]{$L_L$}
    \circledipole[labeloffset=0,tension,tensionoffset=0.7,
        tensionlabel=$U_B$]{\LARGE\textbf{=}}.
    \wire(0,0)(11,0)\qdisk(9,0){2pt}
    \pnode(12.5,5.5){A}\pnode(12.5,0.5){B}
    \tension(A)(B){$u_a$}
\end{pspicture}
```

Usporedimo li električnu shemu nacrtanu pomoću PSTricks biblioteke *pst-circ* i shemu nacrtanu pomoću M4 *Circuit macros*, vidljivo je da je sheme nacrtane pomoću PIC i *Circuit macros* daju puno bolju slobodu u kreiranju shema i novih elemenata, te se drže standarda. Stoga ukoliko se želite više pozabaviti crtanjem, bolje se orjentirati na PIC programiranje.

Kako i sami autori kažu da dodavanje novih elemenata baš i nije jednostavno, korisnika ostavlja bez mogućnosti da s lakoćom kreira macro element po želji.



Slika 5.12: Primjer električne sheme

6 Zaključak

Jedan od osnovnih tehničkih zahtjeva moderne tehničke dokumentacije je standardizacija tehničkih shema. Postoje raznovrsna rješenja pridružena različitim CAD aplikacijama. Međutim, on-line pristup generiranju tehničkih shema posve je zanemaren.

U ovom radu načinilo se on-line rješenje koje zadovoljava nastavne potrebe iz temeljnih kolegija koji se slušaju na usmjerenju Mehatronika i robotika Fakulteta strojarstva i brodogradnje Sveučilišta u Zagrebu, s mogućnošću njegova proširenja na nove sadržaje.

Uz pomoć M4/PIC programa crtaju se električne sheme, blokovi automatske regulacije i programski blokovi. Dobiveno rješenje sprema se u različitim grafičkim formatima rasterske i vektorske grafike ili naredbi prikladnih za LaTeX compiler (PSTricks, Tikz, MetaPost, PNG, GIF, PostScript, EPS, DVI). Pokazana su i druga rješenja crtanja shema tehničke dokumentacije (PsTricks, TikZ).

Za potrebe LaTeX comilacije izveden je modul za Scriptrunner sustav s pripadnim padajućim izbornim ponudama. Korisniku nije potrebno pamćenje naredbi jer ih iz padajućih izbornika jednostavnim klikom može uključiti u editor. Na sličan način realizirane su i sve naredbe ostalih programskih dodataka (Ciruit macro, TikZ, PSTricks,..).

Na koncu napisana su tri različita primjera iz triju različitih područja koja u compiliranom PDF dokumentu imaju interaktivan karakter: korisnik, čitatelj dokumenta, može u primjer unositi vlastite ulazne podatke na temelju kojih će Scriptrunner sustav vratiti odgovarajuće izlazne rezultate. To znači da svaki korisnik čita dokument na svoj način, što je najvažnije značajka dokumenta u edukacijskom

smislu. Primjeri nisu statični, nego dinamički, interaktivni. To predstavlja novost u svijetu generiranja modernih tehničkih dokumenata.

Literatura

[1] Brian W. Kernighan. *PIC — A Graphics Language for Typesetting*, AT&T Bell Laboratories, 1991.

[2] René Seindal, François Pinard, Gary V. Vaughan, Eric Blake. *GNU M4, A powerful macro processor*, Free Software Foundation, Inc., 2007.

[3] Dwight Aplevich. *M4 Macros for Electric Circuit Diagrams in LATEX Documents*
Dohvatljivo sa: http://ece.uwaterloo.ca//Circuit_macros

[4] Tugomir Šurina. *Automatska regulacija*, Školska knjiga, 1991.

[5] Andrew Mertz, William Slough. *Graphics with TikZ*, The PracTEX Journal, 2007, No. 1

[6] John Hobby. *Introduction to MetaPost*
Dohvatljivo sa: http://cm.bell-labs.com/who/hobby/92_2-21.pdf

[7] Uwe Kern. *Extending LATEX's color facilities: the xcolor package*
Dohvatljivo sa: http://www.ctan.org/tex-archive/macros/latex/contrib/xcolor

[8] Till Tantau. *TikZ and PGF, Version 1.01*
Dohvatljivo sa: http://sourceforge.net/projects/pgf/

[9] Keith Reckdahl. *Using Imported Graphics in LATEX and pdfLATEX*, 2006.
Dohvatljivo sa: ftp://ftp.tex.ac.uk/tex-archive/info/epslatex/english/epslatex.pdf

[10] Christophe Jorssen, Herbert Voß†. *A PSTricks package for drawing electric circuits*, 2007.
Dohvatljivo sa: http://mirror.macomnet.net/pub/CTAN/graphics/pstricks/contrib/pst-circ

8 Prilog

8.1. Instalacija M4/PIC u Linux-u

Da bi mogli koristiti M4 i PIC na vašem računalu potrebno je instalirati nekoliko aplikacije. Ovdje je opisan postupak instalacije za Linux operativni sustav, distribucija Debian 4 (Etch).

Kako je već prije napomenuto M4 preprcesor je već u velikom broju slučajeva preinstaliran na Linux operativnom sustavu, ali u slučaju da nije instalira se pokretanjem slijedeće naredbe:

```
apt-get install m4
```

Za PIC compiler ćemo koristiti DPIC - Dwight Aplevich-eva inačica PIC-a, dat je postupak instalacije:

```
cd /tmp
wget http://ece.uwaterloo.ca/~aplevich/dpic/dpic.tar.gz
unzip < dpic.tar | tar xvf -
cd /tmp/dpic
make
cp /tmp/dpic/dpic /bin
```

Ukoliko želite koristiti LaTeXs podrškom za PGF potrebno je pokrenuti slijedeće naredbe:

```
apt-get install tetex-base tetex-extra
```

```
# PGF
apt-get install pgf
```

Za pregledavanje DVI i PS, EPS i drugih formata potrebno je instalirati Ghostscript, pozivajući slijedeću naredbu:

```
apt-get install gs
```

Na poslijetku ako želite koristiti Dwight Aplevich M4 "*Circuit macros*" potrebno je pokrenuti slijedeće naredbe:

```
cd /
wget http://ece.uwaterloo.ca/~aplevich/Circuit_macros/\
Circuit_macros.zip
unzip Circuit_macros.zip
cd /circuit_macros
```

Potrebno je promjeniti sadržaj *homelib.txt* datoteke, to možemo učiniti pomoću *nano* programa:

```
nano homelib.txt
```

zamjenite sadržaj datoteka s:

```
‘define(‘HOMELIB_’,‘/circuit_macros/’)’)
```

gdje je */circuit_macros/* staza gdje se nalaze M4 macro-i. Nadalje,

```
make homelib
make pgfdefault
cp boxdims.sty /usr/share/texmf/tex/latex/
```

Na poslijetku potrebno je ponovno učitati pakete za LaTeX. To ćemo učiniti pomoću slijedeće naredbe:

```
texhash
```

i time završiti postupak instalacije. Sve je spremno za vaš prvi M4/PIC program.

8.2. Instalacija M4 modula u Scriptrunner

Napisani PIC/M4 modul na jednostavan način se može integrirati u Scriptrunner sustav. Sljedeće stavke opisuju postupak instalacije M4/PIC modula u Scriptrunner sustav:

1. kontaktirati *root* korisnika radi dodjele jedinstvenog broja modul
2. jedinstveni broj modula je 1600, broj je potrebno upisati u *sr_plugin_config.php* datoteci koja mora biti uključena u modulu, a dobivna je od *root* korisnika
3. kreirati mapu *m4_1600* unutar *plugins* mape u glavnoj scriptrunner mapi
4. kopirati datoteke modula u *m4_1600* mapu
5. prijaviti se u scriptruner sustav s ovlastima administratora
6. u lijevom izborniku pod *Admin/Global settings* izvršiti naredbu *Register plugins*
7. nakon uspješne instalacije modula postaviti ovlasti za njegovo korištenje

Kao što je vidljivo iz prikazanog postupka, instalacija modula je vrlo jednostavna što pridonosi pisanju novih modula.

8.3. Leksička struktura PIC-a

8.3.1. Slika

Objekt najviše razine u PIC-u je slika ('*picture*'):
picture:

```
.PS optional-width optional-height
     element-list
.PE
```

Ako je parametar *optional-width* naveden, generira se slika toliko incha široka, bez obzira na bilo koju unutarnju dimenziju. Visina slike se umjerava na isti način, osim u slučaju kad je naveden parametar *optional-height*.

8.3.2. Elementi

Popis *element-list* je lista elemenata, a elementi mogu biti:
element:

```
primitive attribute-list
placename : element
placename : position
var = expr
direction
{ element-list }
[ element-list ]
for var = expr to expr by expr do { anything }
if expr then { anything } else { anything }
copy file
copy thru macro
copy file thru macro
sh { commandline }
print expr
reset optional var-list
```

Elementi se odjeljuju novim linijama ili znakom točka-zarez (;). Dugački element može se nastaviti na koncu linije dodavanjem lijeve kose crte (backslash). Komentari se pišu nakon znaka , a završavaju s novom linijom. Imena varijabli počinju malim slovom, a imena pozicijskog mjesta s velikim. Imena mjesta i varijabli zadržavaju svoje vrijednosti iz slike u sliku, ako su napisane u istoj datoteci.

Trenutačna pozicija i smjer gibanja nevidljivog pera (kazala) spremaju se ulaskom u blok ... i ponovno vraćaju u funkciju nakon izlaska iz bloka. Elementi unutar bloka naredbi omeđenih uglastim zagradama [...] promatraju se kao cjelina, čije dimenzije su određene vršnim točkama sadržanog objekta. Imena, varijable i smjer pomaka unutar bloka lokalni su za taj blok i ne prenose se izvan njega.

8.3.3. Temeljni objekti

Slijedi popis temeljnih objekata.
primitive:

```
box
circle
ellipse
arc
line
arrow
spline
move
text-list
```

arrow je sinonim za *line ->*.

8.3.4. Atributi

Popis atributa *attribute-list* je niz od nitijednog ili više atributa. Svaki atribut sastoji se od ključne riječi, nakon koje može slijediti neka vrijednost.

atribut:

```
h(eigh)t expr wid(th) expr
rad(ius) expr diam(eter) expr
up opt-expr down opt-expr
right opt-expr left opt-expr
from position to position
at position with corner
by expr, expr then
dotted opt-expr dashed opt-expr
chop opt-expr -> <- <->
invis solid
fill opt-expr same
text-list expr
```

Sa *expr* je označen izraz, *position* je pozicija, a *corner* je kut ili ugao objekta. Izostavljeni atributi i vrijednosti pune se pretpostavljenim iznosima. Svi atributi se ne primjenjuju na svim temeljnim objektima na isti način. Tako atribut *at* uzrokuje da geometrijski centar bude postavljen na željeno mjesto, dok atribut

with uzrokuje da pozicija na objektu bude postavljena na zadanomjesto. Za linije, spilne-ove i lukove, atributi *height* i *width* odgovarju veličini strijelice. Čisti izraz, bez ključne riječi, uzrokuje pomak u dotadašnjem smjeru.

8.3.5. Text

Tekst je uobičajeni atribut nekog temljnog objekta. Pretpostavljena vrijednost je njegovo postavljanje u geometrijsko središte tog objekta. Dakako, dozvoljen je ispis teksta i bez povezanosti s nekim objektom. Popis *text-list* je lista tekst članova. Tekst član je string omeđen navodnicima iza kojih može doći naredba za pozicioniranje.

text-item:

```
"..." positioning ...
sprintf("format", expr, ...) positioning ...
```

positioning:

```
center ljust rjust above below
```

Ako postoji više tekstovnih članova za neki temeljni objekt, oni se centriraju vertikalno (ako se drugačije ne odredi). Zahtjev za pozicioniranje mora se nezavisno zadati za svaki član.

8.3.6. Pozicije i mjesta

Pozicija je jednostavno neki x,y koordinatni par, koji se može zadati na različite načine.

position:

```
expr, expr
place   expr, expr
place   ( expr, expr )
( position, position )
expr [of the way] between position and position
expr < position , position >
( position )
```

place:

```
placename optional-corner
corner of placename
nth primitive optional-corner
corner of nth primitive
Here
```

Pritom je *optional-corner* jedan od 8 navigacijskih točaka ili središte ili početak, odnosno kraj temeljnog objekta.

optional-corner:

```
.n .e .w .s .ne .se .nw .sw .c .start .end
```

corner:

```
top bot left right start end
```

Svaki objekt u slici ima svoj redni broj koji se *nth* relacijom.

nth:

```
nth
nth last
```

Umjesto *1th*, *2th* i *3th* dakako treba pisati *1st*, *2nd* i *3rd*.

8.3.7. Varijable

Ugrađene varijable i njihove pretpostavljene vrijednosti su:

```
boxwid 0.75          boxht 0.5
circlerad 0.25       arcrad 0.25
ellipsewid 0.75      ellipseht 0.5
linewid 0.5          lineht 0.5
movewid 0.5          moveht 0.5
textwid 0            textht 0
arrowwid 0.05        arrowht 0.1
dashwid 0.1          arrowhead 2
maxpsht 8.5          maxpswid 11
scale 1              fillval .3
```

One se mogu mijenjati u bilo koji trenutak, pa nove vrijednosti ostaju u upotrebi iz jedne slike u drugu, sve dok se ne promijene ili poništе s naredbom *reset*. Varijable promijenjene unutar bloka [i] vraćaju svoje prethodne vrijenosti nakon izlaska iz bloka. Dimenzije se dijele s varijablom *scale* za vrijeme ispisa rezultata na izlaz.

8.3.8. Izrazi

Izrazi u pic jeziku računaju se u aritmetici s pomičnim zarezom. Svi brojevi koji predstavljaju dimenzije uzeti su u inchima.

expr:

```
expr op expr
- expr
! expr
( expr )
variable
number
place .x
place .y
place .ht
place .wid
place .rad
sin(expr) cos(expr) atan2(expr,expr) log(expr) exp(expr)
sqrt(expr) max(expr,expr) min(expr,expr) int(expr) rand()
```

op:

```
+ - * / % ?
< <= > >= == != && ||
```

8.3.9. Definicije (funkcije ili macro naredbe)

Naredbe *define* i *undef* nisu dio gramatike PIC jezika.

define:

```
define(‘name’, ‘replacement text’)
```

undef:

```
undef name
```

Pojavljivanja *$1*, *$2*, itd. u tekstu koji se zamjenjuje, bit će zamijenjena odgovarajućim argumentima ako se *name* pozove sa

```
name(arg1, arg2, ...)
```

Nepostojeći argumenti zamjenjuju se null stringom. Tekst koji se zamjenjuje (*Replacement text*) može imati više linija. The *undef* naredba briše definiciju makro naredbe (funkcije).