Dr Srđan Damjanović
Mr Predrag Katanić

PROGRAMSKI JEZIK

VEE PRO

ELEKTROTEHNIČKI FAKULTET
ISTOČNO SARAJEVO, 2O11.

Recenzent:
Prof. dr Rade Stankić
Prof. dr Petar Bošnjaković

Izdaje:
ELEKTROTEHNIČKI FAKULTET
Istočno Sarajevo

Za izdavača:
Prof. dr Božidar Krstajić

Štampa:
GRAFIKA GOLE
Bijeljina

Tiraž:
200 primjeraka

ISBN: 978-99938-624-7-5

Posveta

Ovu knjigu posvećujem brižnoj supruzi Slobodanki, bogatstvu naše ljubavi Teodori i Mihailu, koji su sa puno razumjevanja omogućili moj rad na ovoj knjizi. Zahvaljujem se takođe mojim voljenim roditeljima Miri i Mirku, za svu nesebičnu ljubav koju su mi pružili.

Srđan Damjanović

S A D R Ž A J

UVOD

VEE Pro je programski jezik prvenstveno namijenjen za programsko povezivanje računara sa *Hewlett Packard* (odnosno danas *Agilent*) mjernim instrumentima. Međutim ovaj programski jezik se uspješno koristi i za programsko povezivanje sa programabilnim mjernim instrumentima ostalih najpoznatijih svjetskih proizvođača. Zbog svoje jednostavnosti u pisanju programa i lakom praćenju toka izvršenja programa, ovaj programski jezik može da posluži početnicima u programiranju za savladavanje prvih koraka u programiranju. Ovaj programski jezik do sada ima više uspješnih verzija. Cilj ovog izdanja je da se prilagodi potrebama studenata u početnoj fazi proučavanja programabilnih mjernih instrumenata, kao i svima onima koji se bave povezivanjem mjernih instrumenta sa personalnim računarima. Izdavanjem ove knjige nadomještena je literatura, koja nedostaje za proučavanje postupka procjene mjerne nesigurnosti iz predmeta Električna mjerenja na Elektrotehničkom fakultetu u Istočnom Sarajevu.

Takođe je evidentan nedostatak sadržaja koji se odnosi na simulaciju laboratorijskih vježbi na računaru iz predmeta Električna mjerenja. Autori su, uvidjevši tu prazninu, brižljivo razradili primjere gotovih programa napisanih u programskom jeziku *VEE Pro*, koji se mogu koristiti za laboratorijske vježbe. Na osnovu toga su studentima i ostaloj stručnoj javnosti ponudili jedan ovakav nastavni sadržaj. Na relativno skromnom broju stranica izložena je jedna konzistentna cjelina pogodna za edukaciju u obrazovnim ustanovama. Knjiga je posebno aktuelna za studente i čitaoce, koji se samostalno bave programiranjem u programskom jeziku *VEE Pro*. Ponuđeni primjeri su, zbog konciznosti izlaganja, djelimično uprošćeni, ali se nevelikim trudom mogu brzo i efikasno dovesti do profesionalne aplikacije. Obrađene oblasti su raznovrsne i brižljivo birane, kako bi čitaoci mogli da uvide višestruke primjene obrađenog materijala u praksi.

Knjiga je napisana zbog nedostatka na tržištu literature na srpskom jeziku za programski jezik *VEE Pro*. Naime, postoje samo uputstva za rad sa raznim verzijama programskog jezika *VEE Pro*, koja su na engleskom jeziku i u elektronskom obliku. To su, uglavnom, obimne dokumentacije izložene na više stotina, čak i oko hiljadu stranica, a uglavnom se odnose na jedan segment problema, koji se obrađuje.

Knjiga je u osnovi podijeljena na osam poglavlja, koja čine tri zasebne cjeline. Prvu cjelinu čine poglavlja od jedan do šest, u kojima su date osnove programskog jezika *VEE Pro*. Drugu cjelinu čini sedmo poglavlje u kome su predstavljeni programi za simulaciju laboratorijskih vježbi na računaru. Treću cjelinu čini osmo poglavlje u kome je opisan postupak procjene mjerne nesigurnosti.

Za svakog programera je bitno da poznaje kako funkcioniše hardver na kome se njegov program pokreće. To je razlog zašto su u prvom poglavlju predstavljene

osnovne hardverskih komponente računara. Zatim je ukratko opisan pojam softvera. Na kraju ovog poglavlja su dati osnovni koraci, koji čine proces pisanja programa.

Postupak instaliranja programa *VEE Pro* predstavljen je u drugom poglavlju. Ukratko su predstavljene skraćenice koje olakšavaju rad sa programom *VEE Pro*. Na kraju je opisano, kako može izgledati postupak pravljenja prvog programa u ovom programskom jeziku.

U trećem poglavlju opisani su osnovni tipovi podataka, koji se koriste prilikom pisanja programa u programskom jeziku *VEE Pro*. Uz svaki tip podatka prikazan je primjer programa, kaji opisuje kako se on koristi.

Paleta menija u programu *VEE Pro* opisana je u četvrtom poglavlju. Detaljno su predstavljeni svi elementi menija, kako bi se što lakše mogli koristiti za pisanje programa. Za pojedine elemente, koji predstavljaju objekte pomoću kojih se sastavlja program, dati su primjeri programa. Predstavljen je i način, kako da se pojedini objekti mogu kombinovati sa drugim objektima u jednu cjelinu, koja se zove program. Opisano je, koji se objekti mogu iskoristili za rješavanje nekih praktičnih problema. Ponuđeni primjeri su djelimično uprošćeni, ali se nevelikim trudom mogu brzo i efikasno dovesti do profesionalne aplikacije. Obrađene oblasti su raznovrsne i brižljivo birane, kako bi čitaoci mogli da uvide višestruke primjene obrađenog materijala u praksi.

U petom poglavlju predstavljena je jedna od dobrih osobina programskog jezika *VEE Pro*, koja daje mogućnost direktnog povezivanja sa dokumentima napravljenim u *Microsoft Office* paketu. U ovom poglavlju predstavljeno je kako se program napisan u programskom jeziku *VEE Pro* može direktno pozivati i koristiti iz programa napisanog u programskom jeziku *Visual Basic*.

Primjeri programa napisanih u programskom jeziku *VEE Pro* predstavljeni su u šestom poglavlju. Prvo su predstavljeni primjeri programa, koji se mogu naći u Helpu programskog jezika *VEE Pro*. Zatim su predstavljeni praktični primjeri kodova programa, koji služe za komunikaciju sa programabilnim mjernim instrumentima.

U sedmom poglavlju su predstavljeni programi, koji se mogu koristiti za simulaciju na računaru labaratorijskih vježbi iz predmeta Električna mjerenja.

Postupak procjene mjerne nesigurnosti opisan je u osmom poglavlju. Ovo poglavlje je dodato u knjigu, kako bi studenti dobili odgovarajuću literaturu iz ove relativno nove oblasti, koja se izučava u okviru predmeta Električna mjerenja. Danas potpuno iskazivanje rezultata mjerenja treba obuhvatiti i informaciju o mjernoj nesigurnosti, kako bi se omogućilo poređenje istih mjerenja izvršenih u različitim institucijama.

Nadamo se da će ova kniga biti od koristi svima onima, koji se bave mjerenjem i koji u svom radu koriste programabilne mjerne instrumente. Sugestije i pitanja mogu se slati na adresu srdamjan@yahoo.com i predrag@telrad.net.

1 OSNOVE PROGRAMIRANJA

Svjedoci smo da živimo u informatičkom dobu i najjednostavnije odluke se ne mogu donijeti, ako se ne posjeduje prava informacija[1]. Za rješavanje kompleksnih problema neophodno je obraditi ogromnu količinu podataka, kako bi se došlo do pravih informacija. Svi ovi poslovi oko dobijanja informacije (prikupljanje, ažuriranje, obrada, prenošenje podataka, itd.) su nezamislivi bez upetrebe računarskih sistema.

Računari su mašine za obradu podataka. One na osnovu određenog skupa ulaznih podataka koje zadaje korisnik generišu odgovarajući skup izlaznih podataka. Ulazni podaci redovno predstavljaju veličine na osnovu kojih se obavlja rješavanje nekog problema, a izlazni podaci predstavljaju rezultate obrade datog problema. Stoga se podrazumijeva da svaki računar posjeduje jedinicu za ulaz (učitavanje) podataka i jedinicu za izlaz (prikazivanje) rezultata. U slučaju kada korisnik računara interaktivno komunicira sa računarom onda je najčešće ulazna jedinica tastatura, a izlazna jedinica ekran. Naravno, postoje i brojni drugi ulazni i izlazni uređaji.

U svakodnevnom životu ljudi upotrebljavaju cifre i slova, koje jednim imenom nazivamo alfanumjerički znaci, za međusobno sporazumijevanje. Za prenos informacija ljudi koriste slike i zvuk, kao i neke posebne znakove. Računar ne radi s decimalnim brojevima. To su uobičajeni brojevi s kojima računamo i brojimo. U računaru ne postoje elektronska kola i sklopovi, koji mogu raditi s tim brojevima. Da bi se u računaru moglo manipulisati informacijama, koje su opisane pomoću cifara, slova, slike, zvuka itd. potrebno ih je binarno predstaviti. Binarni brojevi su, barem na prvi pogled, čudni. Oni postaju uobičajeni i razumljivi tek onda kada se nauči brojanje sa njima i kada se zna njihovo pretvaranje u decimalne brojeve.

Binarni brojni sistem ima bazu dva, što znači da koristi samo skup cifara 0 i 1. Stoga se često naziva i "dualni" sistem brojeva. Koliko god je decimalni brojni sistem značajan i uobičajen za naš svakodnevni život, toliko je i binarni brojni sistem značajan za predstavljanje i shvatanje principa rada računara. Ovo dolazi otuda što elementarne elektronske i magnetne komponente računara u stvari mogu prikazati (tj. zauzeti) samo dva moguća stanja, uključeno (*On*) ili isključeno (*Off*). Primjeri funkcionisanja u binarnom obliku prisutni su i u našem svakodnevnom životu kod električnog zvonca na vratima, sijalice u kući itd. Čak i samo porijeklo riječi ("bis" - dva puta, "binaris" - dvojni) upućuje na bazu dva.

Da bi korisnik mogao da komunicira sa računarom kao mašinom, potrebno je da postoje programi koji to omogućuju. Današnja situacija je takva da iste hardverske komponente mogu izvršavati programe pisane na različitim programskim jezicima,

[1] Srđan Damjanović, Predrag Katanić, Borislav Drakul, *Zbirka zadataka iz poslovne informatike*, Fakultet spoljne trgovine, Bijeljina, 2008, str.17.

da mogu raditi u različitim režimima korišćenja i pod različitim operativnim sistemima. Ovo je dovelo do toga, da pojam računar nedovoljno opisuje kompleksnost ovakvog sistema, pa se često pojam računara uopštava pojmom računarski sistem. Pod pojmom računarski sistem podrazumijevamo složen uređaj-mašinu, čije su mogućnosti u obradi podataka rezultat jedinstvenog djelovanja njenog hardvera i softvera. Ovakav pristup računarskom sistemu nas dovodi do grubog rasčlanjenja računarskog sistema tj. računarski sistem se sastoji iz:

- ***hardvera,***
- ***softvera.***

Drugačije rečeno, jedinstveno djelovanje hardvera i softvera definiše uslove rada i korišćenja računarskog sistema, odnosno definiše okruženje u kome korisnik radi.

Osnovni cilj ovog teksta je da definiše osnovne principe i mehanizme pomoću kojih se jedno takvo okruženje stvara i kako ono funkcioniše. Bolje razumijevanje okruženja, omogućava i efikasnije korišćenje mogućnosti koje ono pruža.

1.1 H A R D V E R

Pod hardverom podrazumijevamo fizičke komponente računarskog sistema kao što su: ***sistemska jedinica*** (kućište računara u koje su smješteni: matična ploča, procesor, memorijski uređaji, napojna jedinica, itd) i ***periferni uređaji*** (ulazni i izlazni). Ulazni periferni uređaji su oni uređaji preko kojih sistem dobija naredbe ili podatke. U računarskom sistemu srećemo ulazne periferne uređaje kao što su: tastatura, "miš", skener, mikrofon, olovka za crtanje i sl. Izlazni periferni uređaji su oni uređaji putem kojih sistem proslijeđuje informacije o izvršenoj naredbi, obrađenim podacima i sl., korisniku sistema. U računarskim sistemima srećemo sljedeće izlazne uređaje: monitor, štampač, ploter, zvučnik, projektor, razni kompjuterski kontrolisani uređaji i sl.

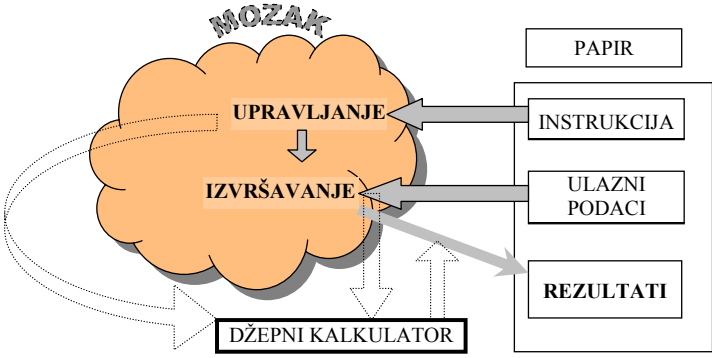
U ovom tekstu ćemo se zadržati samo na sistemskoj jedinici i njenim komponentama koje su ujedno i osnovne komponente računarskog sistema.

1.1.1 Osnovne komponente računarskog sistema

U cilju definisanja osnovnih hardverskih komponenti računarskog sistema i njegovog funkcionisanja, poći ćemo od poređenja ručne i automatske obrade podataka.

U opštem slučaju, ručna obrada podataka odvija se tako što se prethodno na papiru pripremi uputstvo (opis postupka obrade, instrukcije) i podaci sa kojima se počinje obrada. U samom izvršavanju obrade, čovjek čita jednu po jednu instrukciju (korak obrade) sa papira i potom obavlja dvije radnje. Prva radnja je

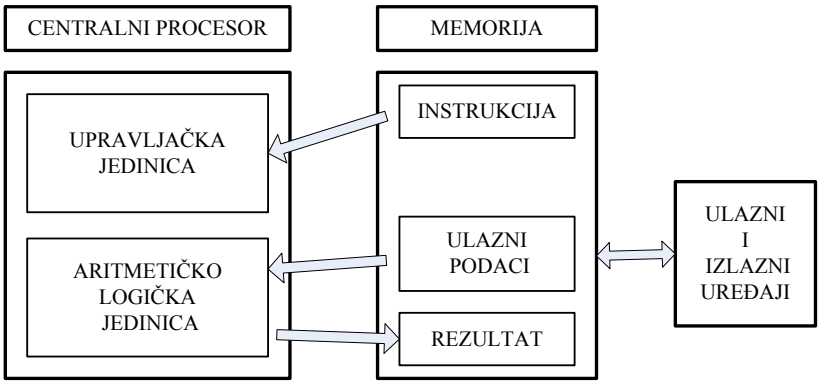
vezana za upravljanje, odnosno tumačenje instrukcije i donošenje odluke o operaciji koja će se izvršiti. Druga radnja je samo izvršavanje operacije, pri čemu se najćešće podaci potrebni za njeno izvršenje čitaju sa papira, a rezultat operacije se takođe zapisuje na papir. Blok dijagram koji predstavlja opisani postupak prikazan je na slici 1.1.



Slika 1.1 Ručna obrada podataka

Na slici 1.1 takođe je prikazana i uloga kalkulatora u obradi podataka, pri čemu se može uočiti da kalkulator može zamijeniti čovjeka samo u izvršenju operacija, a ne i u upravljanju obradom. Naime, čovjek mora da unese podatke u kalkulator i da mu zada operaciju koju treba izvršiti sa tim podacima. Po izvršenju jedne zadate operacije i dobijanju rezultata, kalkulator miruje. Da bi se obrada nastavila, čovjek mora da unese nove podatke u kalkulator i da mu zada novu operaciju itd.

Očigledno, kalkulator nije automat, odnosno mašina koja može da po izvršenju jedne operacije, automatski, bez intervencije čovjeka, pređe na izvršavanje sljedeće operacije. Automatska mašina bi očigledno morala da zamijeni čovjeka, ne samo u izvršavanju operacija sa podacima, već i u upravljanju obradom. Takva automatska mašina jeste sistemska jedinica računara sa komponentama računara koje imaju specijalna zaduženja, tj. takva mašina je računarski sistem ili kraće računar.



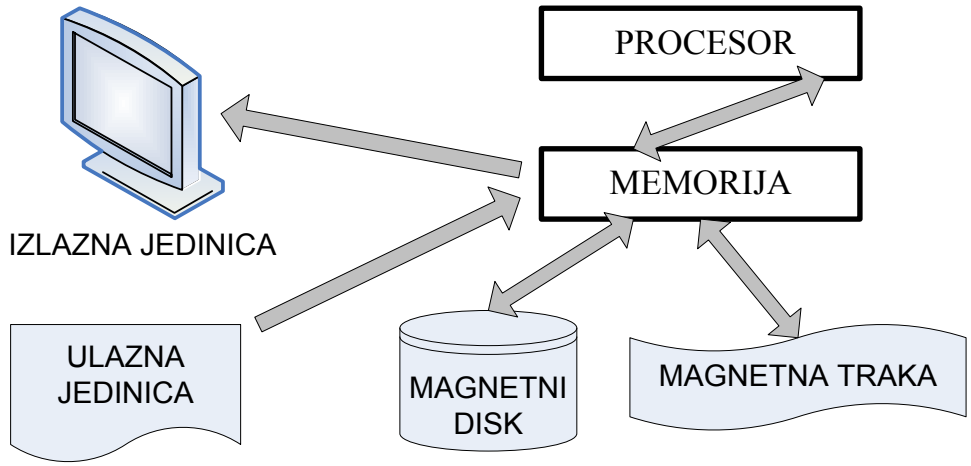
Slika 1.2 Osnovna blok šema računara

Kako je to prikazano na slici 1.2, osnovna komponenta hardvera računarskog sistema je centralni procesor (*central processing unit*) čiji je zadatak da uzima instrukcije iz memorije, analizira ih i potom izvršava. Memorija računara (*memory*) čuva (pamti, skladišti) instrukcije i podatke (početne podatke, međurezultate i rezultate obrade). Ulazni i izlazni uređaji omogućavaju komunikaciju računara s njegovom okolinom. Ulazni uređaji vrše pretvaranje podatak i instrukcija iz forme razumljive čovjeku u formu prilagođenu skladištenju u memoriji računara. Izlazni uređaji pretvaraju rezultate obrade iz forme u kojoj su zapisani u memoriji računaru, u formu razumljivu za čovjeka.

Nešto detaljnija konfiguracija računarskog sistema prikazana je na slici 1.3, na kojoj se može uočiti, da su u cilju bolje preglednosti razdvojeni ulazni i izlazni uređaji, dok su, sa druge strane prikazana i dva nova memorijska uređaja - magnetni diskovi i magnetne trake kao eksterne memorije.

Naime, kapacitet memorije, odnosno, kako se često naziva, operativne memorije, uprkos tome što se stalno povećava, nije dovoljno veliki da uskladišti sve programe i podatke, koji se koriste u automatskoj obradi podataka. Usljed toga, praktično svaka konfiguracija računarskog sistema posjeduje i dodatne memorijske uređaje, koji se nazivaju sekundarnim memorijama i na kojima se trajno čuvaju programi i podaci. Kako u toku rada centralni procesor pristupa samo onim instrukcijama i podacima, koji se nalaze u primarnoj memoriji, programi koji se trenutno izvršavaju i podaci nad kojima se vrši obrada, prebacuju se po potrebi iz sekundarnih memorija u primarnu memoriju.

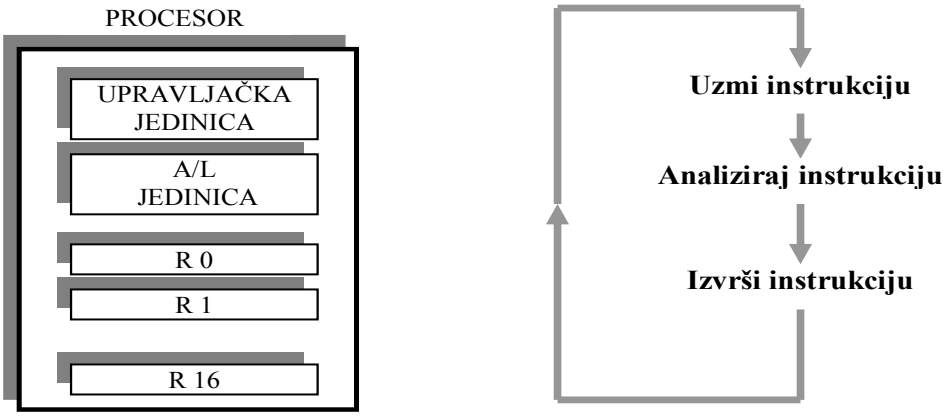
U nastavku izlaganja, kako bismo lakše shvatili suštinu funkcionisanja računarskog sistema za šta je zadužen operativni sistem, ograničićemo se na osnovni opis strukture i funkcije dvije ključne komponente: računarskog sistema - centralnog procesora i primarne memorije.



Slika 1.3 Detaljna konfiguracija računarskog sistema

1.1.2 Centralni procesor

Tipična, iako još uvijek nedovoljno detaljna struktura centralnog procesora, ili jednostavnije procesora, prikazana je na slici 1.4. Osnovna funkcija upravljačke jedinice je da uzima instrukcije iz memorije, analizira ih i i šalje odgovarajuće komande za njihovo izvršavanje ostalim komponentama. Ciklični proces uzimanja, analize i izvršavanja instrukcije, takođe je prikazan na slici 1.4. On se često naziva uzmi-izvrši ciklus (*fetch-execute cycle*).



Slika 1.4 Struktura centralnog procesora

Aritmetičko-logička (A/L) jedinica, kako samo ime govori, sastoji se od elektronskih kola koja obavljaju različite aritmetičke, logičke, kao i neke druge operacije nad podacima. Za A/L jedinicu se kaže da je jedina aktivna komponenta računarskog sistema, u smislu da je jedina komponenta koja može da stvara nove podatke. Jedna od bitnih karakteristika procesora je skup instrukcija koje on može da izvrši. Detaljniji opis tipova intrukcija prevazilazi ovaj kurs.

Pored navedenih komponenti, centralni procesor sadrži i skup posebnih hardverskih jedinica, registara, obično 8 do 16, koji igraju ulogu interne, brze memorije samog procesora. Kako je brzina pristupa registrima približno za red veličine veća od brzine pristupa memoriji, procesor u toku izvršavanja programa u ovim, kako se često nazivaju opštim registrima, čuva podatke kojima mora često da pristupa.

Napredak u razvoju računara je na početku tekao sporo, a zatim je došlo do postepenog ubrzavanja razvoja, da bi danas imali takvu situaciju da se razvoj računarske tehnike ne može ni pratiti tj. gotovo svakodnevno se pojavi novo rješenje u cilju napretka računarskih sistema. Razvoj tehnologije posebno se ogleda na polju razvoja procesora, tako da pojedine komponente procesora poprimaju nove oblike, kao npr. registri procesora koji sada predstavljaju internu memoriju takvih memorijskih dimenzija da omogućavaju mnogo rjeđe pristupe sporijoj *RAM*

memoriji što drastično povećava brzinu rada procesora (interna memorija procesora se naziva "keš" memorija). Današnji računarski sistemi imaju i po desetak mikroprocesora ("srce računara"), koji rade brzinama reda GHz.

1.1.3 Primarna memorija

Primarna (operativna) memorija, ili kraće memorija, je pasivna komponenta računarskog sistema, čija je funkcija da skladišti programe i podatke. Memorija se sastoji od velikog broja ćelija (memorijskih lokacija) jednake dužine, pri čemu svaka ćelija ima svoj redni broj u okviru memorije. Numjerisanje memorijskih ćelija počinje od 0. Očigledno je da redni broj ćelije predstav1ja njenu jedinstvenu adresu. Uvedimo sada predpostavku da jedna memorijska ćelija moža da sadrži jednu i samo jednu instrukciju ili samo jedan podatak. Tada možemo da izvedemo zaključak da svaka instrukcija, odnosno, svaki podatak ima sopstvenu adresu - adresu lokacije u kojoj je smješten.

Uvođenjem koncepta adresa i adresiranja stvoren je bitan mehanizam za funkcionisanje računara, stvorena je mogućnost da se svaka instrukcija i svaki podatak smiješta u određenu memorijsku adresu, da bi mu se kasnije, posredstvom te adrese moglo pristupiti. Pri tome su nad memorijom moguće dvije operacije - *upisivanje i čitanje sadržaja neke adrese*. Realizacija ovih operacija omogućena je posredstvom dva registra, od kojih se jedan naziva adresnim registrom memorije (ARM), a drugi prihvatnim registrom memorije (PRM). Operacija čitanja sadržaja određene memorijske lokacije, obavlja se tako, što se adresa lokacije kojoj se želi pristupiti, upisuje u ARM, a zatim se inicira izvršenje same operacije, da bi se kao rezultat njenog dejstva sadržaj posmatrane lokacije upisao u PRM.

Operacija upisivanja podatka u određenu memorijsku lokaciju odvija se na sličan način. Podatak koji treba da se upiše unosi se u PRM, a adresa lokacije u koju se vrši upisivanje unosi se u ARM, a zatim se inicira operacija upisivanja. Rezultat operacije upisivanja je da se podatak iz PRM upisuje u posmatranu ćeliju, pri čemu se, naravno, njen prethodni sadržaj uništava.

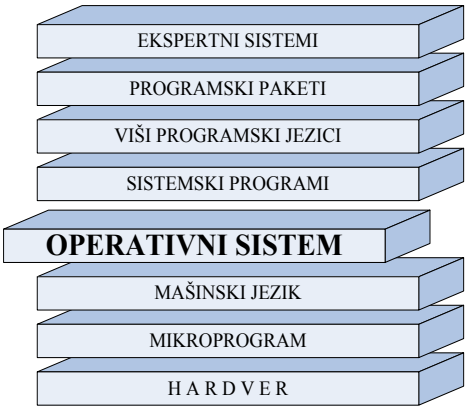
Vrijeme koje protekne od trenutka iniciranja jedne operacije nad memorijom, bez obzira da li je u pitanju čitanje ili upisivanje, do završetka posmatrane operacije, naziva se vremenom memorijskog ciklusa. Jednostavnije rečeno, vrijeme memorijskog cilusa je vrijeme koje mora da protekne između dva uzastopna pristupa memoriji. Pri tome, treba napomenuti da je vrijeme pristupa jednako za sve lokacije operativne memorije, što se razlikuje od situacije kod sekundarnih memorija, gdje vrijeme pristupa nekoj lokaciji zavisi ne samo od fizičke pozicije te lokacije, već i od toga kojoj se lokaciji prethodno pristupilo. Zbog toga se primarna memorija često naziva memorija sa slučajnim pristupom. Ovaj naziv je zapravo doslovan i pomalo nespretan prevod originalnog izraza (random access memory - *RAM*).

1.2 S O F T V E R

Softver (*eng. software*) računara predstavlja skup programskog koda - programa i podataka, smještenih u memoriju računara, koji kao cjelina, povremeno ili stalno, kontroliše i upravljanja radom računara. Ovdje se pod memorijom podrazumijeva sva dostupna memorija računaru (hard disk - HD, flopy disk - FD, CD ROM, DVD medij, traka kao softverski medij itd.). Grubo softver možemo podijeliti na:

- softver operativnog sistema (OS),
- aplikativni softver.

Operativni sistemi su se razvijali sa razvojem računarskih sistema tj. potrebama čovjeka za sve većom količinom informacija. Sa stanovišta OS-a, hardverska komponenta računarskog sistema grubo se može podijeliti u dva dijela, sistemski dio hardvera (procesor, memorija i dr.) i dio hardvera koji čini interfejs sa korisnikom (monitor, tastatura, "miš" i dr.). Za vrijeme pionirskog razvoja računarskih sistema, a samim tim i pionirskog razvoja OS-a, cijena sistemske komponente je bila neuporedivo veća od interfejsa (terminal) tj. rad "mašine" je bio daleko skuplji od rada korisnika. Iz ove činjenice nastaje ideja da prvi operativni sistemi budu tako koncipirani kako bi se omogućilo 100% iskorićenje procesora. Iz ovog ugla OS se posmatra na njegovom najnižem nivou, kada poput nadzornika pazi na disk, procesor, memoriju, štampače i dr.



Slika 1.5 Struktura računara

Slika 1.6 Struktura operativnog sistema

Sa razvojem tehnologije izrade hardvera dolazi do pada cijene sistemskog hardvera, te rad korisnika računara dobija na vrijednosti, pa se ovoj činjenici prilagođava i operativni sistem tj. u prvi plan dolaze odgovornosti OS-a na nivou korisnika. Iz ovog se jasno vidi, da OS ima ulogu posrednika između korisničkih programa i sredstava potrebnih za njihovo izvođenje (hardvera). Ovakva filozofija računarskog sistema je prikazana na slici 1.5, gdje se jasno može vidjeti mjesto i zadatak operativnog sistema.

Kompleksnost operativnog sistema otežava preciznu definiciju. Sa zadržanom uopštenošću, može se reći, da pod operativnim sistemom podrazumijevamo skup sistemskih programa koji djeluju kao posrednik između hardvera i korisnika. Pri tome pruža korisniku usluge, koje olakšavaju projektovanje, implementaciju i održavanje programa, a istovremeno, upravljaju dodjeljivanjem (alokacijom) resursa računarskog sistema u cilju njegovog efikasnog rada.

Kraće rečeno, operativni sistem je organizovan skup sistemskih programa koji upravlja radom različitih komponenti računarskog sistema, sa ciljem da omogući efikasan rad korisnika i efikasno korišćenje resursa samog sistema. Pri tome se poslovi (funkcije) operativnog sistema mogu grubo podijeliti u četiri grupe:

- Upravljanje procesima,
- Upravljanje memorijom,
- Upravljanje uređajima,
- Upravljanje podacima.

Resursi računarskog sistema o kojima OS mora voditi računa su: procesori, memorija, ulazno-izlazni uređaji i podaci. Operativni sistemi mogu biti struktuirani na različite načine: kao monolitni sistemi, kao hijerarhija slojeva, kao virtuelne mašine i kao klijent server model. Jedna od mogućih struktura operativnog sistema prikazana je na slici 1.6. Navedimo neke od poslova koje obavljaju ove komponente operativnog sistema:

1) Jezgro (kernel ili nucleus**)** operativnog sistema obezbjeđuje realizaciju sljedećih funkcija:

- upravljanje sistemom prekida računara i obradu prekida,
- planiranje procesa,
- manipulaciju nad procesima,
- sinhronizaciju procesa,
- komunikaciju među procesima.

2) Upravljanje memorijom podrazumijeva upravljanje operativnom memorijom računara. Obuhvata sljedeće funkcije:

- realizaciju određene strategije dodjele memorije,
- dodjelu memorije,
- realizaciju određene strategije oslobađanja memorije.

3) Upravljanje uređajima obuhvata sljedeće funkcije:

- obezbjeđenje nezavisnosti uređaja,
- obezbjeđenje efikasnosti rada uređaja,
- realizaciju određene strategije dodjele uređaja,
- dodjelu uređaja,
- realizaciju određene strategije oslobađanja uređaja.

4) Upravljanje podacima treba da obezbijedi softverska sredstva za organizovanje i pristup podacima na način koji odgovara korisniku računarskog sistema. Funkcije koje se realizuju na ovom nivou su:

- kreiranje i brisanje fajlova,
- otvaranje i zatvaranje fajlova,
- čitanje i upisivanje,
- upravljanje prostorom na sekundarnim memorijskim jedinicama,
- obraćanje fajlovima po imenu,
- zaštita podataka od namjernog i nenamjernog uništenja,
- zaštita podataka od neovlašćenog pristupa i korišćenja,
- zajedničko korišćenje fajlova.

5) Planiranje obuhvata aktivnosti u vezi sa uvođenjem novih poslova u sistem i određivanje poretka u kojem će se oni izvršavati. Funkcije koje se realizuju na ovom nivou su:

- izbor novog posla za izvršenje,
- dodjela prioriteta poslovima,
- realizacija strategije dodjele resursa.

6) Evidencija obuhvata vođenje evidencije korišćenja resursa sistema po korisnicima i izdavanje računa korisnicima za potrošene resurse.

Očigledno je da akcenat na riječi upravljanje u navođenju funkcija operativnog sistema nije slučajan. Naime, u situaciji, kada se u računaru odvija više aktivnih procesa (programi koji se izvršavaju), jasno je da će ti procesi konkurisati jedan drugom u pogledu korišćenja resursa računara (procesora, memorije, različitih uređaja, datoteka tj. podataka itd.). Zadatak operativnog sistema je da omogući neometano odvijanje svih procesa na takav način, da se svi resursi sistema što efikasnije iskoriste.

Aplikativni ili korisnički programi koriste se na računaru za obavljanje nekih specifičnih poslova. Danas postoji veliki broj aplikativnih programa, koji se razlikuju po namjeni i po veličini. Recimo *Word* se koristi za pisanje i uređivanje teksta, *Excel* za razne matematičke proračune i grafički prikaz podataka, *Access* za pravljene baze podataka itd. Aplikativni programi se mogu pisati u raznim programskim jezicima, ali se svi prepoznaju, u odnosu na druge podatke na računaru, po tome što imaju ekstenziju .EXE.

1.3 PISANJE PROGRAMA

Programi se pišu, kako bi pomogli čovjeku da pomoću računara riješi neki problem. Postoje dva pristupa u razvoju programa[2]:

- od računara ka problemu i
- od problema ka računaru.

U prvom pristupu se polazi od toga da korisnik najprije upozna računar i njegove mogućnosti, a zatim pristupi rješavanju problema uz pomoć računara. U drugom pristupu se prvo nastoji dobro razumjeti problem, zatim napraviti odgovarajući model kako da se taj problem riješi, a tek zatim se pristupa pisanju programa u odgovarajućem programskom jeziku na računaru.

Programiranje u užem smislu predstavlja proces pisanja programa za računar. U širem smislu to je proces pripreme, razrade i pisanja programa radi rješavanja nekog problema na računaru. Proces programiranja zavisi od problema koji se rješava. Međutim, mogu se uvesti neki tipični postupci koji se odvijaju u toku razvoja programa. U programiranju uočavamo sljedeće faze:

- Definisanje problema,
- Izrada algoritma,
- Pisanje i unošenje programa u računar,
- Testiranje programa i ispravke greške,
- Implementacija programa i obuka korisnika i
- Održavanje i nadogradnja programa.

Sve ove korake mora da prati odgovarajuća dokumentacija, u vidu tekstualnih zapisa, grafičkih prikaza i slika.

1.3.1 Definisanje problema

Prva i najbitnija faza u procesu pisanja programa je definisanje problema. Preskakanje ove faze znači sigurno velike probleme u svim narednim fazama koje slijede. U ovoj fazi treba da se uoči problem, određuje se način rješavanja, vrši se analiza postojećeg stanja, analiziraju se iskustva u radu sa ovakvim i sličnim zadacima, biraju metode rada. U ovoj fazi rada često je neophodno definisati fizički i matematički model sistema ili procesa čije ponašanje se želi implementirati na računaru. Fizički model uvijek predstavlja model realnog sistema ili procesa, koji se odvija u prirodi. Matematički model je skup matematičkih postupaka i relacija kao i puteva njihovog rješavanja, koji moraju egzaktno odrediti postavljeni fizički model. Matematički model zapravo predstavlja skup matematičkih relacija koje dovoljno tačno opisuju pojavu ili proces, koji želimo da opišemo programom.

[2] Dr Tihomir Latinović, *Osnove programiranja (Visual Basic)*, Biblioteka Informacione tehnologije, Banja Luka 2007, str. 2.

Najčešće se pri tome koristimo jednačinama i sistemima jednačina, nejednačinama i sistemima nejednačina, različitim transformacijama, funkcijama, vektorima, matricama i drugim matematičkim iskazima.

1.3.2 Izrada algoritma

Ljudi se svakodnevno nalaze u situaciji da rješavaju različite, manje ili više kompleksne, probleme (zadatke). To mogu biti zadaci na radnom mjestu, u kući ili u životnoj sredini, čovjeka uopšte. Čovjek ne bi dorastao ni najobičnijem problemu, da ne koristi razna pomagala tj. alate, mašine uređaje itd. Jedno od pomagala jeste i računar sa svim svojim prednostima. Međutim, iako je računar moćna mašina, još uvijek ne posjeduje razum, tj. ne možemo se potpuno osloniti na njegov automatizam u obavljanju različitih poslova.

Značajna sposobnost ljudi jeste da uoče zadatak, da ga dobro postave, a zatim da ga riješe. Zadatak sadrži skup informacija na osnovu kojih treba izvesti izvjesne zaključke, koji predstavljaju rješenje zadatka. Ovakav skup informacija čini *ulazne veličine zadatka*, a rješenja zadatka zovu se *izlazne veličine zadatka*.

Izbor informacija koje čine ulazne veličine zadatka, kao i tačno formulisanje samog zadatka možemo zvati *postavka zadatka*. Jasno je da prije nego što se pristupi rješavanju, mora se izvršiti tačna postavka zadatka. Rad na postavci zadatka zahtijeva dobro poznavanje oblasti kojoj pripada zadatak.

Kada je izvršena postavka zadatka može se pristupiti rješavanju istog. Uređen skup pravila koja se formuliše u cilju rješavanja zadatka zove se *algoritam*. Ulazne veličine zadatka zovu se ulazne veličine algoritma, a izlazne veličine zadatka zovu se izlazne veličine algoritma. Svako pravilo, iz skupa pravila formulisanih u cilju rješavanja zadatka, zove se algoritamski korak. Prema tome, algoritam se sastoji od niza algoritamskih koraka, kroz koji se vrši transformacija ulaznih veličina algoritma, sve dok se ne dođe do izlaznih veličina algoritma tj. rješenja. U ovakvom nizu algoritamskih koraka mora se znati prvi i zadnji algoritamski korak, a za sve ostale algoritamske korake jednoznačno je određen sljedeći algoritamski korak. Veze između algoritamskih koraka određuju *strukturu algoritama*.

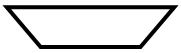
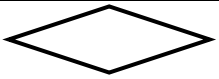
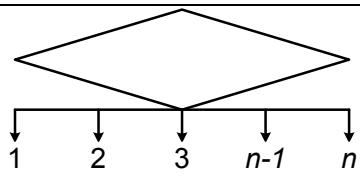
1.3.2.1 Grafički zapis algoritma

Algoritmi za rješavanje pojedinih problema mogu da budu veoma složeni[3]. U takvim slučajevima algoritmi sa tekstualnim opisom pojedinih koraka dosta su nepregledni. Zato je za zapis algoritama pogodno koristiti njihovo grafičko prikazivanje. Kod ovakvog prikazivanja algoritama svaki algoritamski korak je prikazan grafičkim simbolom, koji su između sebe povezani linijama saglasno

[3] Dr Lazar Miličević, Mr Lazar Radovanović, *Programiranje (Visual Basic)*, Ekonomski fakultet, Brčko, 2005, str.11.

strukturi algoritma. Oblik grafičkog simbola za algoritamski korak ukazuje na funkciju koju obavlja određen algoritamski korak. U tabeli 1.1 dati su grafički simboli za pojedine algoritamske korake i opisana njihova funkcija.

Tabela 1.1 Grafički simboli algoritamskih koraka

Grafički simbol algoritamskih koraka	**Funkcija algoritamskog koraka**
početak/kraj	Ukazuje na prvi ili zadnji algoritamski korak
	Definiše ulazne veličine algoritma
	Definiše izlazne veličine algoritma
	Definišu obradu podataka
	Uslovni algoritamski korak
	Definiše višestruku odluku
	Povezivanje ili konektor
	Tok algoritma
	Dvosmjerni prenos informacija

Grafički zapis algoritma zove se *blok-šema algoritma*. Ovakav zapis algoritma odlikuje se sljedećim osobinama:

- preglednije praćenje toka algoritma,
- omogućuje zapis algoritma na način, koji obezbjeđuje lako otkrivanje grešaka u strukturi algoritma,
- omogućuje kraći i jasniji zapis algoritma, nego pisanim jezikom,
- daje preglednu vezu između detalja i cjeline algoritma, i
- algoritam zapisan u obliku blok-šeme nezavisan je od kasnijeg korišćenja algoritma u odnosu na hardversku podršku i programski jezik u kome će program biti pisan.

Posljednja osobina je posebno značajna, jer grafičko prikazivanje algoritma nije orijentisano na prenošenje algoritma na računar što omogućuje da algoritam u obliku blok-šeme mogu koristiti i osobe koje ne poznaju računare i programiranje. Detaljisanje algoritma, pri grafičkom prikazivanju može biti različito i zavisi od namjene blok-šeme algoritma. Blok-šema algoritma treba da je toliko detaljna da svaki algoritamski korak bude razumljiv za onog ko će koristiti algoritam. Za složenije algoritme pogodno je koristiti i blok- šeme različitog nivoa detaljisanja algoritamskih koraka.

Ako se za prikaz algoritma koriste različiti nivoi detaljisanja algoritma, onda se koristi opšta blok-šema i detaljna blok-šema algoritma. Opšta blok-šema algoritma sadrži algoritameke korake koji predstavljaju veće funkcionalne ili logičke cjeline u složenom algoritmu. Detaljna blok-šema algoritma sadrži algoritamske korake u kojima je jasno naznačena funkcija svakog algoritamskog koraka. Opšta blok-šema služi za uvid u logičku strukturu složenog algoritma, a detaljna za uvid u sve detalje algoritma sa gledišta njegovog izvršavanja.

1.3.2.2 Struktura algoritma

Veze izmedju algoritamskih koraka u algoritmu definišu strukturu algoritma. Strukture algoritama, na koje se može razložiti proizvoljna struktura algoritma, zovu se elementarne strukture algoritama.
Elementarne strukture algoritama su:

- linijska struktura,
- ciklična struktura.

Ove strukture se medju sobom bitno razlikuju sa gledišta izvršavanja algoritma.

Linijska struktura algoritma

Niz algoritamskih koraka u kojem se svaki algoritamski korak može izvršiti najviše jedanputa, u toku jednog izvršavanja algoritma čini linijsku strukturu. Prema tome, karakteristika linijske strukture algoritma jeste da se poslije izvršenog jednog algoritamskog koraka može preći samo na algoritamski korak koji nije prethodno izvršen, u toku jednog izvršavanja algoritma. Linijska struktura algoritma može biti prosta ili razgranata.

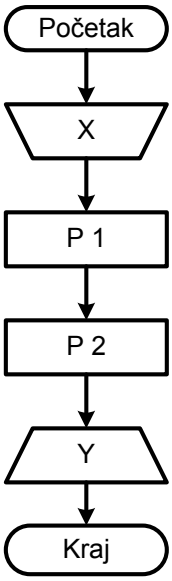
Prosta linijska struktura

Prosta linijska struktura algoritma je ona linijska struktura kod koje se svaki algoritamski korak izvršava jedanput u toku jednog izvršavanja algoritma.

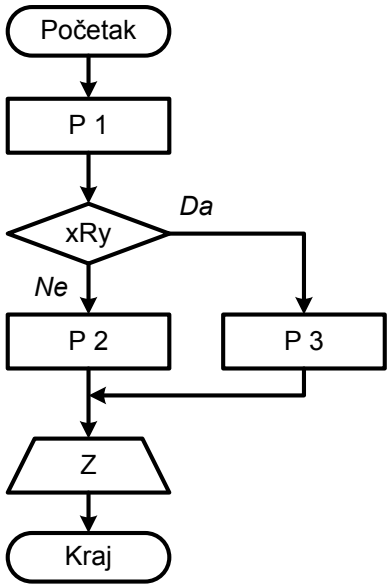
Algoritmi sa prostim linijskim strukturama, sastoje se isključivo od algoritamskih koraka ulaza, obrade ili izlaza. U ovakvim algoritamskim strukturama redoslijed izvršavanja algoritamskih koraka ne zavisi od ulaznih veličina, niti od među rezultata u procesu izvršavanja algoritma. Primjer ove strukture prikazan je na slici 1.7.

Razgranata linijska struktura

Razgranata linijska struktura je ona linijska struktura algoritma, kod koje se svaki algoritamski korak izvršava najviše jedanput u toku jednog izvršavanja algoritma. To znači da u razgranatoj algoritamskoj strukturi postoje algoritamski koraci koji se jedanput izvrše, ali postoje i algoritamski koraci koji se ne izvrše u toku jednog izvršavanja algoritma. U razgranatim algoritamskim strukturama mora postojati barem jedan uslovni algoritamski korak, koji omogućuje grananje algoritma.



Slika 1.7 Prosta linijska struktura



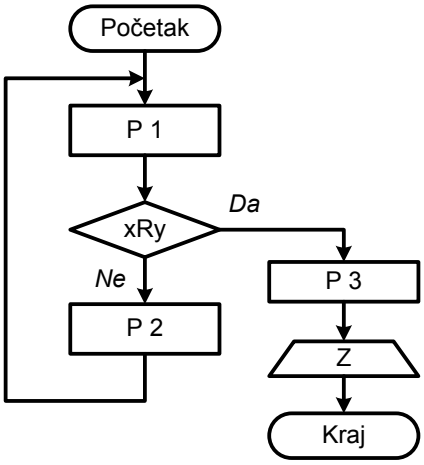
Slika 1.8 Razgranata linijska struktura

Elementarna razgranata struktura može se komponovati od 3 proste linijske strukture i jednog uslovnog algoritamskog koraka. Na slici 1.8 prikazana je elementarna razgranata struktura sa prostim linijskim strukturama P_1, P_2 i P_3 i uslovnim algoritamskim korakom xRy, gdje su x i y prethodno definisane veličine, a R relacija između veličina x i y. Relacija R može biti ispunjena i tada se prelazi na prostu linijsku strukturu P_3 ili neispunjena i tada se prelazi na prostu linijsku strukturu P_2. Važno je uočiti da se pri izvršavanju algoritma, na slici 1.8, izvršava uvijek jedna od prostih linijskih struktura P_2 ili P_3, u zavisnosti od relacije xRy.
Veličine x i y mogu biti definisane sa ulaza ili to mogu biti među rezultati određeni u prostom linijskom programu P_1. Relacija između veličina x i y odgovara prirodi ovih veličina. Ako su x i y brojne veličine može se ispitivati da li su jednake ili koja je od ovih veličina manja, odnosno veća. Kasnije ćemo vidjeti da ove veličine mogu biti i nizovi.

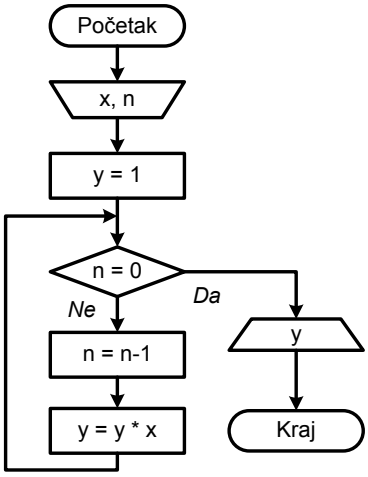
Ciklična struktura algoritma

Niz algoritamskih koraka u kojem se jedan ili više algoritamskih koraka može izvršiti više od jedanput, pri jednom izvršavanju algoritma, predstavlja cikličnu strukturu ili ciklus.

Ciklična struktura može biti konstantna i promjenljiva ciklična struktura. Svaka od ovih, cikličnih struktura u najopštijem obliku sastoji se od dvije proste linijske strukture P_1 i P_2 (*slika 1.9*) i uslovnog algoritamskog koraka xRy. Ako je relacija navedena u uslovnom algoritamskom koraku ispunjena vrši se izlazak iz ciklusa, a ako uslov nije ispunjen prelazi se na prostu linijsku strukturu P_2, poslije čega se ciklus ponavlja. Relacija koja definiše izlazak iz ciklusa zove se *izlazni kriterijum ciklusa*. Naravno, da se može ciklus organizovati i tako da se izlazak iz ciklusa vrši ako uslov, koji definiše izlazni kriterijum, nije ispunjen, a ciklus nastavlja ako je uslov ispunjen.



Slika 1.9 Konstantna ciklična struktura



Slika 1.10 Izračunavanje stepena

Konstantna ciklična struktura

Ciklična struktura algoritma u kojoj ne dolazi do promjene zakona obrade u algoritamskim koracima koji čine ciklus, zove se konstantna ciklična struktura. Izlazni kriterijum kod konstantnih cikličnih struktura najčešće je broj ponavljanja ciklusa ili dostignuta tačnost pri računanju po interativnim postupcima. Ukažimo samo da kod iterativnih ciklusa broj prolazaka kroz ciklus nije unaprijed poznat.

Primjer 1.1: Sastaviti algoritam za izračunavanje stepena $y = x^n$
gdje je x nepoznata, a stepen može imati vrijednosti $n = 0,1,2,...$

Ovdje su x i n ulazne veličine, a y izlazna veličina. Pošto je eksponent cio broj, stepen se može izračunati uzastopnim množenjem. Prema tome, algoritam za rješavanje ovog zadatka sadrži ciklus u kojem će se vršiti $n - 1$ množenje osnove x, a izlazni kriterijum je broj izvršenih množenja (slika 1.10).

Važno je uočiti da je u ovom zadatku vrijednost *n* promjenljiva (zadaje se kao ulazna veličina) te je nemoguće algoritam za ovaj zadatak sastaviti kao linijsku algoritamsku strukturu. U ovom zadatku korišćen je simbol, koji predstavlja operaciju dodjeljivanja vrijednosti promjenljivoj. Uočimo da ovaj simbol određuje proces izračunavanja vrijednosti na lijevoj strani, a zatim ovako izračunata vrijednost se dodjeljuje promjenljivoj na desnoj strani simbola. Zapis *" y = x "* čita se "*vrijednoet x dodjeljuje se promjenljivoj y* ". Treba imati u vidu da ovaj simbol opisuje proces formiranja vrijednosti *x*, a zatim dodjeljivanje ove vrijednosti promjenljivoj y. Ovo znači da na mjesto *x* može stajati i aritmetički izraz, čiji argument može biti i vrijednost *y*.

Tako je u algoritmu, u ovom primjeru, korišćen zapis *n = n - 1* što znači da se vrijednost promjenljive n umanjuje za jedan i tako formirana vrijednost dodjeljuje kao nova vrijednost promjenljive *n*.

Algoritam na slici 1.10 ima konstantnu cikličnu strukturu, jer zakon obrade u svim algoritamskim koracima, koji se nalaze u ciklusu, ne mijenja se za vrijeme izvršavanja algoritma.

Promjenljiva ciklična struktura

Ciklična struktura u kojoj dolazi do promjene zakona obrade, u jednom ili više algoritamskih koraka, koji se nalaze u ciklusu zove se promjenljiva ciklična struktura. Promjena zakona obrade može se odnositi na promjene operacija koje vrše obradu informacija ili na promjene promjenljivih koje učestvuju u pojedinim algoritamskim koracima.

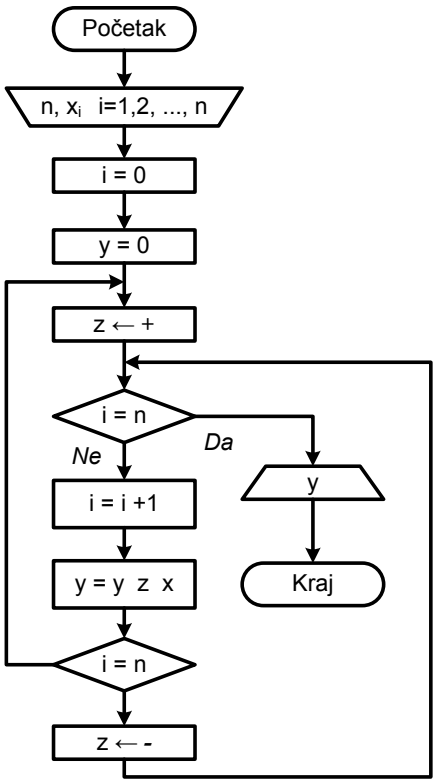
Primjer 1.2: Sastaviti algoritam koji izračunava sumu

$$y = \sum_{i=1}^{n} (-1)^{i+1} x_i \qquad (1.1)$$

Rješenje: Ovdje su ulazne veličine u algoritam $n, x_1, x_2, ..., x_n$, a izlazna veličina y. Razvijanjem, suma (1.1) se može zapisati u obliku

$$y = x_1 - x_2 + x_3 - ...$$

Ovo izračunavanje je u algoritmu sa slike 1.11 riješeno tako da se operacija *z* mijenja naizmjenično (+ ili -).



Slika 1.11 Promjenljiva struktura

Prvo je to operacija sabiranja (+ $\Rightarrow z$), a zatim operacija oduzimanja (- $\Rightarrow z$), pa opet sabiranja itd. Prema tome, u ovom algoritamskom koraku zakon obrade se mijenja u toku izvršavanja algoritma, naizmjenično sabiranje i oduzimanje. Pored toga, u istom algoritamskom koraku dolazi do promjena promjenljivih koje učestvuju u obradi, to su redom promjenljive $x_1, x_2, \ldots, x_n$. Ciklusi u kojima postoje algoritamski koraci sa ovakvim vrstama promjena, u toku izvršavanja algoritma, su promjenljive ciklične strukture.

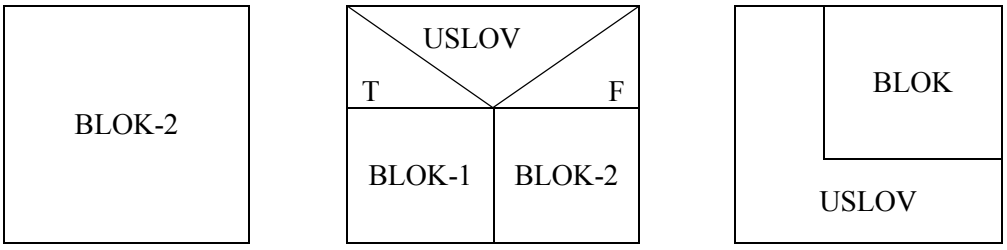
1.3.2.3 Struktuirani dijagrami toka

Standardni dijagrami toka su definisani međunarodnim standardom *ANSI X3.5*. Dosta su zastarjeli, ali se još uvijek često koriste za prikazivanje algoritama. Međutim, problemi kod standardnih dijagrama toka nastaju kada treba da se shvati globalna struktura predstavljenog algoritma.

Na primjer:

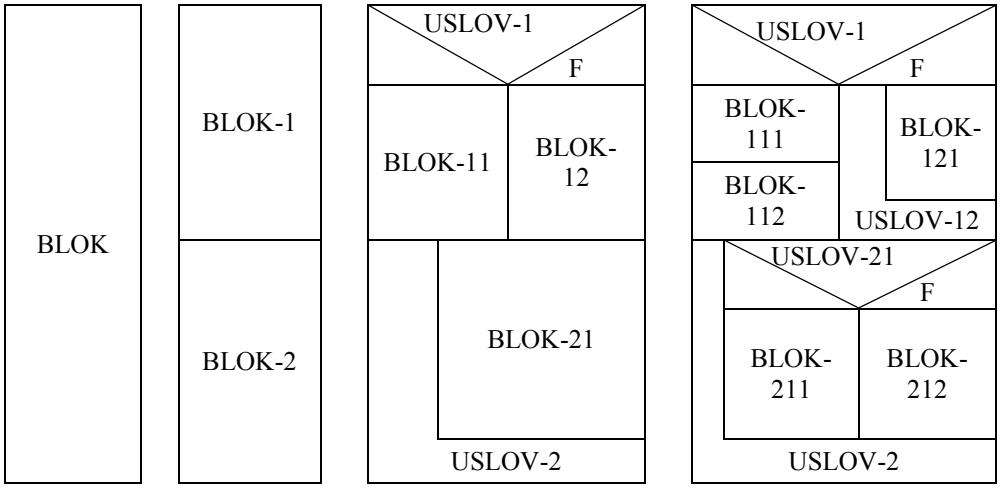
- koliko ima ciklusa (zatvorenih putanja) u dijagramu,
- u kakvom se odnosu nalaze ti ciklusi,
- kom ciklusu pripada koji blok, itd.

Struktuirani dijagrami toka su novijeg datuma i podržavaju moderne principe struktuiranog programiranja. Struktuirano programiranje se zalaže za algoritme i programe sa jasno definisanom i lako razumljivom strukturom. Pored strukture programa, posebna pažnja se posvećuje i strukturi obrađivanih podataka. Jedan od važnih elemenata struktuiranog programiranja je i razvoj algoritma ili programa u koracima preciziranja. Osnovni cilj je ušteda programerskog truda u sastavljanju, ispitivanju i prepravljanju kako sopstvenih tako i tuđih programa. Ovaj cilj se ponekad postiže većim utroškom potrebnog memorijskog prostora za podatke i programe, a ponekad i sporijim izvršavanjem dobijenog programa. Smatra se da je, u današnje vreme sve moćnijih i relativno jeftinijih računara, čovjekov trud najskuplji i da to treba najviše da se štedi. Na slici 1.12 prikazane su osnovne vrste elemenata struktuiranih dijagrama toka. Svi elementi su pravougaonog oblika sa ulazom na vrhu i izlazom na dnu. To omogućuje laku dekompoziciju bilo kog bloka dijagrama na preciznije korake.



Slika 1.12 Elementi struktuiranih dijagrama toka

Elementi struktuiranih dijagrama toka predstavljaju logičke cjeline koje se nazivaju osnovnim upravljačkim strukturama. Tako, *selekcija* predstavlja vrlo čestu strukturu, uslovno izvršavanje jednog od blokova BLOK-1 ili BLOK-2 zavisno od toga da li je USLOV ispunjen (T) ili nije (F). Pored ove osnovne selekcije, postoje i druge vrste selekcija. One su prikazane, zajedno sa korišćenim oznakama u struktuiranim dijagramima toka. Druga česta upravljačka struktura je *ciklus* što podrazumijeva izvršavanje BLOK-a sve dok se ne ispuni USLOV. Postoje i druge varijante ciklusa.



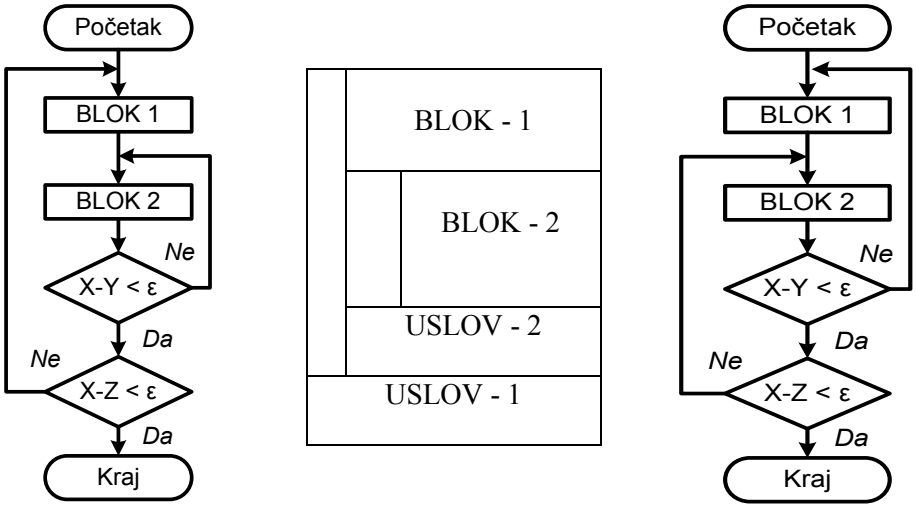
Slika 1.13 Pravljenje složenog struktuiranog dijagrama

Na slici 1.13 prikazan je primjer formiranja složenog struktuiranog dijagrama toka. U prvom koraku cio dijagram je jedan blok koji predstavlja cio algoritam. U drugom koraku taj blok je podijeljen na dva uzastopna bloka *BLOK-1* i *BLOK-2*. U trećem koraku *BLOK-1* je pretvoren u selekciju sa uslovom *USLOV-1* i uslovno izvršavanim blokovima *BLOK-11* i *BLOK-12*. Istovremeno *BLOK-2* je pretvoren u ciklus sa sadržajem *BLOK-21* i uslovom završetka *USLOV-2*. Na kraju, u četvrtom koraku *BLOK-11* je podijeljen na dva uzastopna bloka, *BLOK-12* je pretvoren u ciklus, a *BLOK-21* je pretvoren u selekciju.

Dobra osobina struktuiranih dijagrama toka je sprečavanje sastavljanja nepreglednih algoritama. Pored toga, struktuirani dijagrami toka su kompaktniji. Na istoj površini može da se iskaže više informacija nego pomoću standardnih dijagrama toka. Mana im je, ako to smije da se uzme kao mana, što se koristi više različitih elemenata nego kod standardnih dijagrama toka i za njihovo razumjevanje nije dovoljna samo intuicija.

Na slici 1.14 *a)* prikazan je standardni dijagram toka, koji se sastoji od dva uklopljena ciklusa, a na slici 1.14 *b)* prikazan je ekvivalenti struktuirani dijagram toka. Standardni dijagram toka na slici 1.14 *c)* se, na prvi pogled, neznatno razlikuje od dijagrama na slici 1.14 *a)*. Međutim, kada se bolje pogleda, uočava se

da dva ciklusa u dijagramu više nisu uklopljena jedan u drugi. Oni se djelimično preklapaju. Ovakva struktura algoritma ne može da se iskaže struktuiranim dijagramom toka.



Slika 1.14 Pravljenje struktuiranog dijagrama

1.3.3 Pisanje i unošenje programa u računar

Algoritam zapisan u obIiku blok-šeme ne može biti prihvaćen i izvršen od strane računara. Da bi algoritam bio prihvaćen i izvršen od strane računara mora biti zapisan na način, koji obezbjeđuje određen nivo detaljizacije algoritma, što ne mora biti u blok-šemi algoritma.

Algoritam zapisan tako da je prihvatljiv od strane računara zove se *program*, a proces pisanja programa zove se *programiranje*.

Da bi program mogao da se izvrši na računaru mora biti zapisan na mašinskom jeziku računara. Međutim, pisanje programa na mašinskom jeziku predstavlja takvu detaljizaciju algoritma, da je to vrlo težak posao za čovjeka, a uz to i vrlo podložan greškama. Pored toga, pisanje programa na mašinskom jeziku zahtijeva poznavanje konstruktivnih osobina računara, što takođe onemogućuje da se veći broj ljudi obuči za ovaj posao.

Da bi se omogućilo efikasnije pisanje programa počeli su se stvarati posebni jezici za komunikaciju između čovjeka i računara. Tako su razvijeni *simbolički* i *programski jezici*. Programi zapisani na ovim jezicima mogu se pomoću posebnih programa za prevodjenje tj. *prevodioca*, prevesti na mašinski jezik i zatim izvršiti na računaru.

Programski jezici su vještački jezici konstruisani za pisanje progama, pri čemu, po pravilu nije potrebno poznavanje konstruktivnih osobina računara. Važno je

uočiti da su programski jezici algoritamski orijentisani, tj. oni omogućuju korisniku da zapiše algoritam za rješavanje određenog zadatka pomoću računara. Prema tome, da bi jedan zadatak riješili na računaru moramo prije svega sastaviti algoritam, koji opisuje rješenje zadatog problema.

1.3.4 Testiranje programa i ispravke greške

Kada se program napiše potrebno je prvo uraditi njegovo testiranje. Prilikom testiranja programa otkrivaju se greške u pisanju programa. Greške mogu da budu sintaktičke i logičke. Sintaktičke greške nastaju prilikom kucanja naredbi u programu. U programskom jeziku *VEE Pro*, to su naredbe koje se kucaju u gotovim objektima od kojih se sastavlja program. Najčešće greške su izostavljanje nekog slova ili da se neko slovo otkuca dva puta. Ove greške se puno lakše otkrivaju od logičkih grešaka. Logičke greške nastaju zbog pogrešnog rasporeda objekata u programu, tako da program ne može da se završi do kraja ili se na kraju dobije rezultat koji nije dobar. Da bi se otklonile ove greške, prvo treba dobro razmotriti napisani algoritam. A upravo jedna od velikih grešaka programera je da su počeli da pišu program, a da nisu prije toga napravili algoritam. Drugi način otklanjanja logičkih grešaka je da se u programu postave kontrolne tačke. Zatim da se program izvršava postepeno od jedne do druge kontrolne tačke, pri čemu se provjeravaju vrijednosti promjenljivih i izlazi svakog objekta. Treći način otklanjanja logičkih grešaka je da se program izvršava korak po korak, od jednog do drugog objekta, korišćenjem opcije *Step Into*. Na ovaj način se i najlakše dolazi do otkrivanja grešaka, a zatim i ispravljanja grešaka u programu.

Kada se program u potpunosti istestira, potrebno je napraviti izvršnu verziju programa preko opcije *Create Run Time Version* u glavnom meniju *File*. Na ovaj način se od razvojne verzije programa koja ima ekstenziju "*.VEE*", pravi izvršna verzija programa koja ima ekstenziju "*.VXE*". Korisnik bi trebalo da radi samo sa izvršnim verzijama programa. Ako korisnik radi sa razvojnom verzijom programa tada vrlo lako može da pokvari program pomjeranjem ili brisanjem samo jedne linije za povezivanje objekata. Sa druge strane pisac programa štiti svoj autorski rad isporučivanjem korisniku samo izvršne verzije programa.

1.3.5 Implementacija programa i obuka korisnika

Tek kada se program u potpunosti istestira, on se daje korisniku na korišćenje. Prije nego što se program da korisniku na korišćenje, svaki programer bi trebao da napravi detaljno uputstvo za korišćenje programa. Uputstvo treba da sadrži slike svih ekrana, koji se mogu pojaviti u toku izvršenja programa, a za svaki objekat na slici treba dati njegove funkcije, tip i raspon ulaznih (izlaznih) podataka. Uputstvo može biti napravljeno u papirnoj ili elektronskoj formi, ali najbolje i u jednoj i drugoj formi. Pored uputstva programer mora da korisniku pruži obuku u korićenju

isporučenog programa. Ako se radi o programima za komunikaciju sa programabilnim mjernim instrumentima preko *USB/GPIB* interfejsa, koji se na računar spaja na *USB* port, posebnu pažnju treba obratiti na grešku koja se može pojaviti, kada računar automatski ne prepozna ovaj interfejs. Ova greška se otkriva tako što na *USB/GPIB* interfejsu svijetle crvena kontrolna svjetla. A ova greška se otklanja vrlo lako, tako što se ovaj interfejs odspoji i ponovo spoji na *USB* port, nakon čega na njemu treba da svijetle zelena kontrolna svjetla.

Proizvodnja programa[4] za tržište je djelatnost slična proizvodnji svih drugih vrsta proizvoda. Kupac obično očekuje slične uslove kupoprodaje, kao u slučajevima kupovine uobičajenih tehničkih proizvoda. Međutim, program je veoma specifičan proizvod i njegova zaštita nije nimalo jednostavna. Pravni aspekti proizvodnje i korišćenja programa predstavljaju u opštem slučaju delikatnu, prilikom sporova komplikovanu i ponekad kontroverznu oblast. Klasična prodaja programa je dosta rijetka pojava. Pod "klasičnom prodajom" programa podrazumijevamo u stvari kupovinu programa i prava na njegovu preprodaju. Drugim riječima kupovina programa je "klasična kupovina" onda kada je slična kupovini nekretnina ili automobila. Tada kupac ima potpuno pravo raspolaganja svojim vlasništvom i može zatim da ga po volji proda, pokloni, ili uništi. Kupovinu programa praktikuju državne organizacije i/ili kompanije (vlada, vojska, policija, elektrane, banke, itd.) gdje program može bitno da utiče na sigurnost rada ili poslovanja, a za njih je jeftinije da kupe programe nego da ih sami razvijaju.

U pravnim državama piratsko kopiranje programa je jednako težak prekršaj kao i povreda patentnih prava ili piratsko kopiranje i distribucija knjiga, muzičkih i video diskova. Ako bi neko neovlašćeno lice uzelo na primjer neku knjigu i bez odobrenja izdavača i autora počelo da tu knjigu samostalno kopira i kopije stavlja u prodaju, onda bi to bio prekršaj i pravno i moralno. Treba razumjeti da zakon koji štiti izdavača i autora štiti u stvari najviši društveni interes. Ako bi proizvođaču knjige, ili programa, njegov proizvod bio nekažnjeno piratski ukraden, onda bi kao direktna posljedica toga opao interes za proizvodnju tako riskantnog proizvoda. Društvo bi neminovno bilo suočeno sa usporenim razvojem ili nazadovanjem. To je razlog što neovlašćeno korišćenje softvera nije samo pravno nedopustiv akt, već predstavlja i ozbiljno kršenje profesionalnog morala jer potkopava temelje zdravih odnosa u okviru programske profesije.

1.3.6 Održavanje i nadogradnja programa

Šaljiva izreka kaže da „svi programi sadrže greške dok se ne dokaže suprotno, što je inače nemoguće“. Imajući u vidu da „u svakoj šali ima i po malo istine i šale“, dolazimo do toga da je veoma teško pri proizvodnji programa pružiti bilo

[4] Dr Jozo J. Dujmović, *Programski jezici i metode programiranja*, Akademska misao, Beograd, 2003, str 3.98 do 3.104.

kakvu garanciju. Tačnije proizvođači programa umjesto garancije za svoje proizvode najčešće daju izjavu da ne prihvataju bilo kakvu odgovornost za primjenu i posljedice primjene svog softvera.

Čak i softveri koji služe za rješavanje nekih matematičkih problema, koji imaju izuzetnu vrijednost i mogu se koristiti sa visokim stepenom pouzdanosti daju ovakve izjave. Ipak, uvijek postoji mogućnost da se, i pored vrlo male vjerovatnoće, desi slučaj da se pri nekim ulaznim parametrima pojavi greška u nekom programu. I ako bi se taj program koristio u proračunu ili u radu nekog veoma osjetljivog uređaja (tipičan primjer su letjelice i njihovi sastavni djelovi), moglo bi doći do fatalnih posljedica za koje proizvođač softvera ne može prihvatiti odgovornost. Stoga se i daje izjava o ne prihvatanju odgovornosti.
Izjave o ne prihvatanju odgovornosti postale su uobičajena pravna forma i primjenjuje se čak i onda kada to, na prvi pogled, ne izgleda neophodno.

Teško je zamisliti kakve bi to fatalne posljedice mogle da nastupe kao rezultat greške u programu za obradu teksta, a da pri tome autor teksta ili njegov izdavač ne budu krivi, nego da svu krivicu snosi programer koji je pisao program ili kompanija koja program distribuira. Naravno, može se dogoditi da zbog greške u programu neki podatak u tekstu neželjeno promjeni vrijednost i da to ima neke neprijatne posljedice, ali normalno bi bilo je da je kriv „onaj ko je koristio nož na naopak način, a ne proizvođač noža“. Poenta je međutim u tome da program za obradu teksta ne može da se izvršava u „apsolutnoj izolaciji“. On upisuje podatke po disku i može da greškom dovede do direktnog ili poslijedičnog gubitka nekih korisničkih programa ili podataka, pa se proizvođač softvera sa razlogom odlučio na sličnu formulaciju kao i autori biblioteke numjeričkih programa.

Imajući u vidu da većina programa radi u okruženju drugih programa i podataka i da se šteta drugim podacima i/ili programima ipak može dogoditi i pored najveće pažnje u programiranju, slijedi da je u ovim slučajevima uputno koristiti izjavu o nepriznavanju garancije. Tim prije što su korisnici softvera navikli da programski proizvodi u određenom malom procentu sadrže greške, tako da potpunu garanciju i ne očekuju. Garancija se stoga u većini slučajeva odnosi samo na besplatnu zamjenu medijuma sa programima, koji nisu čitki zbog grešaka na medijumu.

Druga forma garancije koju proizvođači unikatnog ili maloserijskog softvera ponekad nude kupcima je garancija za interventno održavanje u ograničenom periodu. Ova garancija ima za cilj da obezbjedi da proizvođač softvera u ograničenom periodu (najčešće godina dana) bude u obavezi da po reklamaciji kupca izvrši sitnije popravke i dorade isporučenog programskog proizvoda.

Sa druge strane, imajući u vidu izuzetnu lakoću i brzinu kojom se i veoma veliki programski proizvodi mogu neovlašteno iskopirati, proizvođači programa redovno pokušavaju da svoj proizvod zaštite od toga. Pa pored pravne zaštite putem „*copiryght*“ koriste i razne tehničke norme zaštite od neovlašćenog kopiranja njihovih proizvoda. Postoji veliki broj metoda kako se to može postići, a neke od njih se zasnivaju i na specijalnom hardveru. Jedan od oubičajenih načina

da se obezbjedi da određeni program može da radi isključivo na određenom računaru, sastoji se u tome da se serijski broj procesora (koji se programski može očitati) dostavi proizvođaču softvera, koji zatim taj broj upisuje (najčešće kriptografisano) u program koji treba da radi na tom računaru. U toku rada programa, program povremeno testira da li je serijski broj procesora isti kao i očekivani serijski broj, pa ako to nije slučaj prestaje sa ispravnim radom. Naravno, u tom slučaju pravni aspekt korišćenja je drugačije koncipiran od uobičajene „copyright" zaštite. Sada princip licence za korišćenje softvera nije više „jedna kopija jedan korisnik" (na bilo kom mjestu i na bilo kom računaru), već „jedna kopija, jedan računar". Pri tome broj simultanih korisnika takvog softvera može, ali ne mora, da bude ograničen. Osnovna ideja proizvođača softvera je obično da njegova zarada treba da bude proporcionalna koristi koju korišćenjem softvera ostvaruje njegov korisnik.

Proizvođači softvera (naročito ako se radi o unikatnom softveru) često korisnicima nude ugovorno održavanje softvera. Pod održavanjem se obično podrazumijevaju popravke i dorade koje se rade na zahtjev korisnika, koji je otkrio grešku ili nedostatak u softveru koji koristi, a za koje ima ugovorno održavanje. Ugovorom o održavanju se moraju precizno specificirati uslovi održavanja. To najčešće može biti:

1) vremenski rok u kome proizvođač softvera treba da otkloni otkrivene greške i nedostatke, ili
2) obaveza da se na zahtjev kupca angažuje određeni stručnjak proizvođača, koji će određeno vrijeme provesti radeći u pravcu poboljšanja datog softvera, ali obično bez obaveze da se ispune neki specifični zahtjevi. Budući da je proizvođač softvera spreman na određene intervencije u softveru na zahtjev kupca, to se ponekad može smatrati kao određena forma garancije.

Primjeri navedene forme održavanja softvera obuhvataju obično intervencije u dva osnovna pravca. Korisnik u datom periodu (npr. u toku godine dana) može da detaljno ispita programski sistem sa mnogo širim spektrom kombinacija ulaznih podataka od onoga što je mogao da uradi proizvođač prilikom razvoja programskog proizvoda. Tom prilikom postoji mogućnost da korisnik identifikuje takve kombinacije ulaznih parametara za koje programi daju neispravne rezultate. Obaveza proizvođača se svodi na to da modifikuje program tako da se navedene neispravnosti otklone. Drugi pravac se sastoji u tome, da se modifikuje korisnička forma, odnosno onih dijelova koji definišu tip i formu unosa ulaznih podataka od strane korisnika. Korisnik koji rutinski koristi program duže vremena redovno otkrije sitne nepraktičnosti u načinu komuniciranja, podatke koji se ponavljaju, nepotrebna ograničenja, testove i slično. Tako da ima razloge da pokrene zahtjev za manjim izmjenama programa. Treba naglasiti da pod ovaj vid modifikacije ne

spadaju suštinske modifikacije algoritamske prirode, jer bi to iziskivalo pravljenje nove verzije programa.

Druga vrsta ugovora o održavanju softvera, koja se primjenjuje u domenu sistemskog softvera, ne obuhvata interventno održavanje, već periodičnu isporuku novih verzija instaliranog softvera. U takvim slučajevima proizvođač ima ekipu programera, koji permanentno dorađuju i usavršavaju postojeći softver. Kada se broj intervencija u postojećem softveru nakupi, tako da se nova verzije softvera po svojim osobinama u dovoljnoj mjeri razlikuje od predhodne, onda se ta nova verzija automatski distribuira korisnicima, koji imaju ugovor o održavanju softvera.

Gotovo svaki program je doživio neke modifikacije, a razvoj programa ima obično svoju nomenklaturu u vidu oznake v.m koja označava:

- „verziju“ (v) i
- „modifikaciju“ (m) datog programa.

Kada programer razvija prototip programa onda je to „nulta verzija“, koja može imati veliki broj raznih modifikacija dok se ne dobije kompletan proizvod, koji prođe završno testiranje i spreman je za tržište. Takav proizvod se najčešće označava kao verzija 1.0 (skraćeno V1.0).

Veoma je važno da se u toku razvoja softvera vodi uredna i precizna evidencija o modifikacijama koje su u toku. Vremenom se nagomila veliki broj listinga programa od kojih mnogi mogu biti zastarjeli, ali se to ne može otkriti ako na njima ne stoji odgovarajuća oznaka. Prema tome, u toku razvoja i testiranja programa potrebno je poslije svake modifikacije programa unijeti na programima izmjenjenu oznaku. Počinje se sa V0.1, pa zatim svaka modifikacija ima naredni broj (V0.2, V0.3, itd.) dok se ne dođe do izlaska na tržište sa verzijom V1.0. Zatim se tokom razvoja softvera distribuiraju verzije V1.1, V1.2, itd. Po pravilu poslije manje od 10 modifikacija prelazi se na verziju V2.0. Podrazumijeva se, da se ovaj proces odvija saglasno sa potrebama tržišta, pa V2.0 treba da obuhvati pored popravki grešaka i proširenja, koja čine značajnu razliku u odnosu na verziju V1.0. Zatim se isporučuju modifikacije V2.1, V2.2 itd. Neki popularni programi doživjeli su veći broj verzija, ali rijetko više od 10.

Lako je uvidjeti da ne postoji nikakva čvrsta definicija i ujednačeni pogled na to nakon koliko modifikacija treba da se pojavi nova verzija programa. Tako da označavanje brojeva modifikacija i verzija u priličnoj mjeri zavisi od mjere, ukusa i tradicije koja je svojstvena nekom proizvođaču programa. Ipak, relativno visoki broj verzija i modifikacija programa predstavlja neku formu priznanja da posmatrani programski proizvod uspješno živi na tržištu i da po svojoj prilici ima razvijenu korisničku bazu.

2 UVOD U PROGRAMSKI JEZIK *VEE PRO*

Hewlett Packard je već dugo jedan od vodećih proizvođača mjerne opreme i mjernih instrumenata visoke klase tačnosti. Može se reći da većina metroloških laboratorija u čitavom svijetu ima bar jedan njihov instrument. Kompanija *Hewlett Packard* je jedna multinacionalna kompanija, koja se bavi i proizvodnjom računara, računarske opreme i softvera. Zato se prije par godina iz ove kompanije izdvojila kompanija pod nazivom *Agilent*, koja se specijalizovala za proizvodnju mjerne opreme, mjernih instrumenata, mjerno-akvizicionih sistema i programa za podršku u procesu mjerenja i obradi rezultata mjerenja. Koliko je *Hewlett Packard* značajan u proizvodnji mjernih instrumenata, govori i podataka da je GPIB interfejs za povezivanje mjernih instrumenata sa računarom postao standard, koji danas koriste i ostali vrhunski proizvođači mjerne opreme. Ovaj interfejs je omogućio, da se na vrlo jednostavan način istovremeno poveže sa računarom više programabilnih instrumenata. Tako su nastali i prvi sistemi za automatsko etaloniranje, koji su činili kalibratori i mjerni instrumenti koji su se etalonirali.

Sa početkom proizvodnje programabilnih mjernih instrumenata *Hewlett Packard* je počeo da pravi i specijalizovane programe, koji su služili za uspostavljanje dvosmjerne komunikacije između računara i mjernih instrumenata. Među svim tim programima najbolje se pokazao programski jezik *VEE Pro*.

VEE Pro je linijski programski jezik prvenstveno namijenjen za programsko povezivanje računara sa *Hewlett Packard* (odnosno danas *Agilent*) mjernim instrumentima, ali i programabilnim mjernim instrumenata od ostalih najpoznatijih svjetskih proizvođača. Ovaj programski jezik do sada ima više uspješnih verzija (*VEE 3.0, VEE 4.0, VEE 5.0, VEE 6.0, VEE 7.0, VEE 8.0 i VEE 8.5*). Ovaj programski jezik se preporučuje onima koji su savladali elementarne osnove bar nekog drugog programskog jezika, na primjer *C++*, *Visual Basic, Pascal, Delfi* itd. Zbog svoje jednostavnosti u pisanju programa i praćenju toka izvršenja programa, ovaj programski jezik može da posluži i početnicima za savladavanje prvih koraka programiranja. Nadam se da će ova knjiga bar malo doprinjeti tome.

Osnovne prednosti upotrebe programskog jezika *VEE Pro* su:

1) Jednostavnost do krajnjih granica je jedna od osobina, koja ga izdvaja od ostalih razvojnih programskih paketa.
2) Veliki broj gotovih i raznolikih objekata za upravljanje.
3) Funkcionalan i raznolik grafički interfejs za unos i prikaz podataka.
4) Virtuelni generatori funkcija, impulsa i šuma sa promjenljivom amplitudom i frekvencijom.
5) Podržava komunikaciju sa svim programima iz *Windows* okruženja, tako da se ne gubi vrijeme na programiranje onoga što *Windows* već sadrži.
6) Mogućnost povezivanja sa svim *Hewlett Packard* (odnosno danas *Agilent*) mjernih instrumentima, ali i programabilnim mjernim instrumenata od ostalih najpoznatijih svjetskih proizvođača.

2.1 PRIMJENA RAČUNARA U METROLOGIJI

Bez obzira na mnogobrojne uspješno izvedene automatizacije mjernih procesa[5] u vremenu prije primjene računara, može se sa pravom reći da doba automatizacije u mjerenjima započinje sa primjenom modernih računara. Posmatrajući električne i neelektrične veličine, bilo je sasvim logično da se automatizacija mjerenja zasnovana na primjeni računara široko razvije, prvo kod mjerenja električnih veličina, s obzirom da je tu mjerna informacija sadržana u električnom signalu. Sa druge strane, dodatni uslov za razvoj automatizacije mjerenja neelektričnih veličina bio je nastanak nove generacije mjernih pretvarača neelektričnih u električne veličine.

Posmatrajući računare koji se koriste u mjernim procesima uočava se da se oni mogu podijeliti po namjeni na dva osnovna tipa i to:

- računare posebne namjene ugrađene u mjerni instrument i
- računare opšte namjene.

Kod mjernih instrumenata, koji koriste ugrađeni računar posebne namjene, prisustvo računara, najčešće mikroračunara nema značaja za operatera, jer on nema mogućnost pristupa programskom rezidentu računara, pa ni mogućnost izmjene programa. Kod takvih računara program se generalno izrađuje u mašinskom kodu i smješta u *ROM* ili *PROM*.

Računari opšte namjene, koji se dodaju mjernim instrumentima u mjernim procesima i procesima upravljanja zasnovani su na mikroračunarima i miniračunarima, a programirani su u jezicima visokog nivoa. Oni su otvoreni prema operatoru tako da on ima mogućnost da modifikuje zadatak koji računar treba da preuzme.

U prvim danima primjene računara u mjerenjima, u vojnom vazduhoplovstvu, on je posmatran samo kao upravljački i kontrolni element. Od tada, se njegova uloga stalno transformisala, šireći se do danas u poznatih 5 uloga računara u mjerenjima, a to su:

1) Da obezbjeđuje pristupačnu i moćnu mašinu za računanje, sposobnu da za potrebe mjernog procesa obavlja sva potrebna izračunavanja od jednostavnih algebarskih operacija, pomoću statističke analize rezultata mjerenja do složenih procjena spektra Furijeove transformacije.

2) Da upravlja mjernim postupkom, tj. da upravlja mjernim instrumentima i uređajima od prve do posljednje predviđene mjerne tačke, tj. od puštanja mjernog sistema u rad do okončanja mjernog postupka, kao i istovremeno, da prikuplja rezultate mjerenja i podatke o parametrima sredine u kojoj se obavlja mjerenje. Ovom ulogom računar povezuje pojedinačna mjerna sredstva u jedinstveni

[5] Petar Pravica, Ivan Bagarić, *Metologija električnih veličina opšti deo*, Nauka, Beograd, 1993, str.247 do 252.

automatizovani mjerni sistem, sa mogućnošću optimizacije metroloških karakteristika mjernog sistema kao cjeline.
3) Da memoriše i čuva rezultate mjerenja na magnetskim trakama, disketama, ili diskovima velikog kapaciteta. Pun doprinos ove uloge računara se dostiže samo kada se u punoj mjeri koristi naredna uloga računara.
4) Da obrađuje rezultate mjerenja primenjujući prikupljene mjerne podatke o parametrima sredine i unaprijed poznate korekcije, koristeći matematički aparat, a posebno matematičku statistiku, specifične biblioteke podataka, postupke određivanja optimalnih pravih i krivih, a imajući u vidu unaprijed zadate relacije, propisane nacionalnim ili međunarodnim standardima.
5) Da proizvodi dokumenta, kao što su izvještaj o mjerenju ili atest o kalibraciji, tabelarne preglede, rezultata mjerenja, grafičke prezentacije rezultata u vidu dijagrama, histograma itd.

Imajući sve uloge računara u vidu, pojavljuje se kao jedan od najznačajnijih faktora u primjeni računara upravo interakcija između ovih pet uloga, tj. funkcija. Zaista, ako tok naučne aktivnosti od prikupijanja i ispitivanja podataka, postavljanja hipoteze, projektovanja rješenja, njegove analize i optimizacije, kroz dokumentaciju i nazad ka mjerenju, treba da bude aktivno pomognut od strane računara, onda pomenuta interakcija mora neophodno da postoji. Univerzalnost računara je karakteristika, koju ne posjeduje nijedno drugo tehničko sredstvo, i predstavlja njegovu veliku prednost, koja mu omogućuje opisane uloge u mjernom procesu.

Već iz izložene uloge računara u metrologiji, jasno je da je njegova primjena dovela do ogromnih promjena u mjernoj tehnici, posebno kada su u pitanju električne veličine. Primjena računara u metrologiji dala je jedan novi kvalitet procesu mjerenja koji se ogleda u sljedećem.

1) Proširene su mogućnosti metrologije, jer je automatizovanje mjerenja primjenom računara omogućilo da se izvode i ona mjerenja koja ranije nije bilo moguće obaviti, s obzirom na zahtjev da se u kratkom vremenskom intervalu izvrši veoma veliki broj mjerenja na velikom broju mjernih tačaka.

2) Povećan je kvalitet mjerenja smanjenjem ukupne mjerne nesigurnosti s obzirom na sljedeće:

a) uticaj subjektivnog faktora pri mjerenjima sveden je na najmanju mjeru, isključivanjem, mogućih grubih grešaka pri očitavanju, zapisivanju i obradi rezultata mjerenja,

b) ostvarena je skoro potpuna identičnost postupka mjerenja, što je značajno poboljšalo ponovljivost mjerenja, i time smanjilo slučajnu komponentu mjerne nesigurnosti kod višestruko ponovljenih mjerenja,

c) po pravilu, povećan je broj mjernih tačaka, kao i broj ponovljenih mjerenja po jednoj mjernoj tački u odnosu na klasična mjerenja, što takođe smanjuje slučajnu komponentu mjerne nesigurnosti,

d) omogućena je automatska primjena korekcija, čime je smanjen broj neisključenih sistematskih grešaka, a sa time i sistematska komponenta mjerne nesigurnosti,

e) omogućeno je izvođenje mjernog postupka izvan radnog vremena, kada su smanjene smetnje svih vrsta, a posebno noću kada su optimalni uslovi za mjerenje,

3) Smanjeni su troškovi mjerenja, jer je:

a) skraćeno vrijeme mjerenja, a kompletni izvještaji o mjerenju se dobijaju odmah po završetku mjernog postupka,

b) smanjeno angažovanje stručnog kadra, s obzirom da veći dio mjernog postupka može da se odvija bez kontrole operatora, ili pod kontrolom niže stručnog kadra,

c) produženo raspoloživo radno vreme na 24 časa dnevno.

4) Omogućena je automatizacija kalibracije, što je ubrzalo i podiglo kvalitet kalibracije klasičnih etalona i mjernih sredstava, i omogućilo kalibraciju programabilnih etalona i mjernih sredstava.

5) Mjerni postupak i postupak obrade rezultata mjerenja nisu razdvojeni, već se, šta više obavljaju simultano, sa mogućnošću da se mjerni rezultati prikazuju odmah, ili naknadno putem reprodukovanja sa zapisa.

Proizvođači mjernih instrumenata još uvijek nude na tržištu sve tri generacije mjernih instrumenata i to: analogne, digitalne i programabiIne mjerne instrumente. Analogni mjerni instrumenti koristeći davno uspostavljene, jednostavne ali pouzdane postupke, mjere sve postojeće fizičke veličine, prikazujući ih u preglednom, i u nekim situacijama još uvijek najpogodnijem obliku (instrumenti u pilotskoj kabini, automobilu, komandnoj sali elektrane, itd,). Digitalni mjerni instrumenti nastali su zahvaljujući razvoju poluprovodničke tehnologije i zahtjevima korisnika mjerne opreme, i brzo su se pokazali tačnijim i preciznijim u odnosu na analogne mjerne instrumente. Rješavanjem problema spoljnog upravljanja (daljinskog upravljanja) digitalnim mjernim instrumentima nastali su programabilni mjerni instrumenati.

Kao i kod uvođenja svake nove tehnologije, primjena računara u metrologiji zahtijeva značajne dodatne materijalne troškove i napore za savlađivanje potrebnih stručnih znanja iz ove oblasti. Nedostaci primjene računara se mogu svesti na slijedeće:

1) Dodatni troškovi nabavke računarske opreme i nabavke ili izrade potrebnih programa. Treba imati u vidu, da je mjerno mjesto u najsavremenijim svjetskim metrološkim laboratorijama opremljeno sa tri računana, gdje prvi stalno upravlja mjernim postupkom, drugi preuzima od prvog mjerne podatke, obrađuje ih i štampa izlazne dokumente, dok na trećem, operator usavršava postojeću mjernu metodu ili iznalazi nova rješenja, provjerava dobijene mjerne informacije i priprema nove oblike izlaznih dokumenata.

2) Neophodnost kompatibilnosti korišćenih mjernih sredstava računarskoj opremi, odnosno prilagođenost svih jedinica mjernog sistema istom interfejsu.

3) Neophodnost zamjene starijih generacija mjernih instrumenata novim programabilnim mjernim instrumentima, posebno onih iz grupe mostova, kompenzatora, potenciometara, komparatora i sličnih.

4) Dodatno stručno znanje operatora iz oblasti računarske tehnike.

U poređenju sa prednostima, ovih par sitnih nedostataka je zanimarivo u odnosu na prednosti koje računari donose. Zato se u budućnosti očekuje sve veća primjena računara u metrologiji.

2.2 INSTALIRANJE *VEE PRO* PROGRAMA

Program *VEE Pro* se može instalirati na računar automatski, kada se ubaci instalacioni disk ili korišćenjem *Setup* aplikacije. Prvo se pojavi ekran prikazan na slici 2.1. Na njemu se može izabrati instaliranje tri programa: *Agilent VEE Pro, Agilent VEE Runtime* i *Agilent VEE Express*. Program *Agilent VEE Pro* se koristi za pisanje novih programa. Program *Agilent VEE Runtime* se koristi za pravljenje izvršne verzije programa, koja ima ekstenziju **.vxe**. Izvršna verzija programa se može poslije pokretati nezavisno od programa *VEE Pro*. Program *Agilent VEE Express* posjeduje gotova rješenja za podešavanje programabilnih mjernih instrumentima i prikupljanje rezultata mjerenja.

Slika 2.1 Instaliranje VEE Pro programa

Kada se pokrene postupak instaliranja *VEE Pro* program i ostali dijelovi programskog paketa instaliraju se na hard disk. Nije dovoljno samo prekopirati datoteke sa instalacionog diska na hard disk računara. Mora se upotrijebiti Setup aplikacija, koja će dekomprimirati i instalirati sve potrebne datoteke na odgovarajuće direktorijume. Pošto je *VEE Pro* licenciran softver potrebno je imati šifru, kako bi mogli da nastavimo proces instaliranja programa. Šifra se unosi u polje *Product Key*. Prije instaliranja ovog programa potrebno je provjeriti da li računar zadovoljava minimalne zahtjeve za instalaciju i rad programa. Da bi utvrdili minimalne zahtjeve računara potrebno je na instalacionom disku pročitati datoteku *Readme*.

Da bi se ovaj program mogao koristiti za povezivanje računara sa programabilnim instrumentima, potrebno je uz ovaj program dodatno instalirati na računaru i odgovarajuće biblioteke koje se nalaze na posebnom instalacionom disku (*Agilent IO Libraries*).

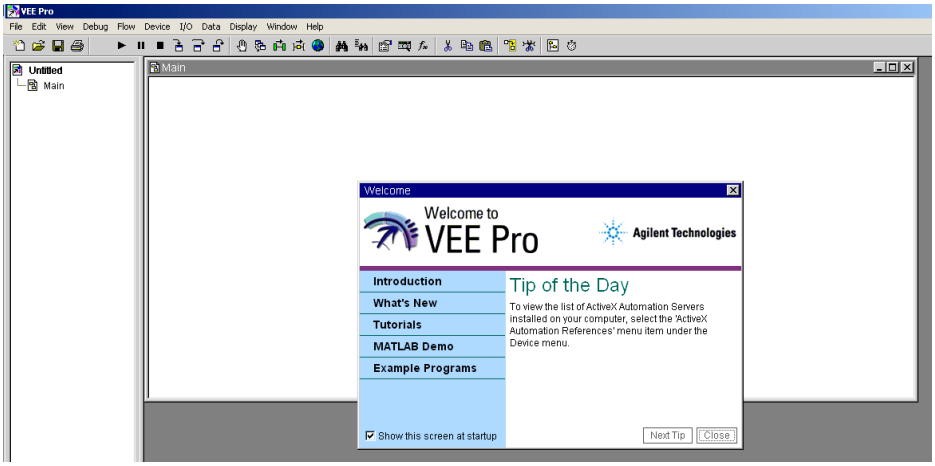
2.3 POKRETANJE *VEE PRO* PROGRAMA

Aktiviranje programa *VEE Pro* u *Windows* operativnim sistemima vrši se kroz Start meni (*Start→Programs→VEE Pro)*. Naravno, da ovo nije jedini način da se pokrene program *VEE Pro*. Kao i za bilo koji drugi program u *Windows* operativnim sistemima postoji više načina za njihovo pokretanje npr. može se napraviti ikona-prečica (*Shortcut*) pa preko nje pokretati program.

Pokretanjem *VEE Pro* programa, odmah se uočava da prozor programa pripada grupi standardnih prozora u *Windows* okruženju, samim tim njegovo radno okruženje čini naslovna linija sa dugmadima za automatsko upravljanje veličinom prozora, linija menija, palete sa alatima, radni prostor, klizači za vertikalno i horizontalno listanje sadržaja dokumenta i statusna linija.

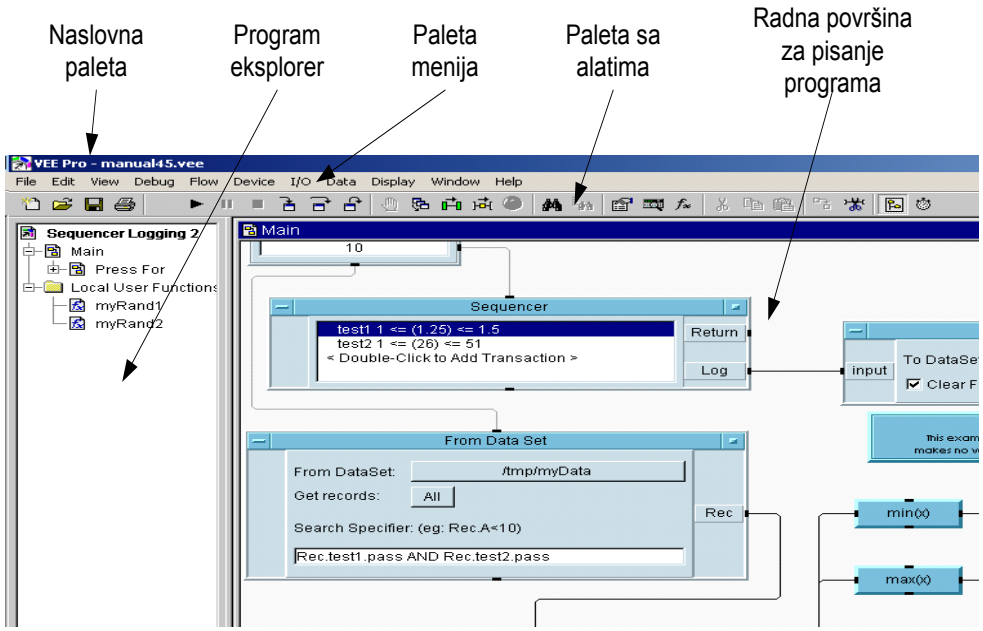
Nakon pokretanja programa pojavljuje se prozor prikazan na slici 2.2. Na njemu se može izabrati jedna od sljedećih opcija ako je potrena neka pomoć u radu:

- ***Instruction*** - ako nikada prije nije korićen ovaj program da se sazna nešto o osnovnom konceptu rada sa objektima u ovom programu.
- ***Whats New*** - ako se želi saznati šta ovaj program pruža novo u odnosu na druge slične programe.
- ***Tutorijal*** - ako se želi pregledati multimedijalno uputstvo za rad sa ovim programom koje se nalazi na instalacionom disku.
- ***MATLAB Demo*** - ako se želi raditi sa *MATLAB Demo* programom koji pruža više mogućnosti za grafički prikaz rezultata.
- ***Example Programs*** - ako se žele otvoriti gotovi primjeri sa već urađenim programima.



Slika 2.2 Početni ekran nakon pokretanja programa VEE Pro

U slučaju kada ste već ranije radili sa ovim programom ili želite da otvorite prethodno napravljen program tada je potrebno kliknuti na dugme ***Close***. Nakon toga se pojavljuje prozor prikazan na slici 2.3. Tu je prikazana radna površina na koju se mogu dodavati objekti za naš novi program. U slučaju da se želi otvoriti već postojeći program, potrebno je otići na paletu menija na ***File*** pa ***Open*** i iz već poznatog okruženja (kao i kod ostalih programa *Windows* okruženja) iz eksplorer menija izabrati program koji želimo da pokrenemo.



Slika 2.3 Radna površina za pisanje programa

2.4 SKRAĆENICE U PROGRAMU *VEE PRO*

Program *VEE Pro* je dosta specifičan i različit u odnosu na ostale klasične ili objektno orjentisane programske jezike, kod kojih se sve naredbe pišu preko koda. U ovom programskom jeziku većina naredbi piše se preko gotovih objekata, koji se samo podešavaju za izvršenje određenih naredbi. U ovom programskom jeziku novi programi se prave tako što se objekti dodaju na radnu površinu, pa se onda spajaju linijama, koje predstavljaju tok izvršavanja programa. Da bi se olakšalo i značajno ubrzalo pisanje programa korisno je poznavati skraćenice na tastaturi, koje se mogu koristiti u programu. Bez upotrebe skraćenica na tastaturi često je praktično i nemoguće uraditi izmjene u programima. Ovo pogotovo dolazi do izražaja u onim programima koji imaju puno objekata, kada svi objekti ne mogu da stanu na jedan ekran. Osnovne skraćenice koje se koriste u programskom jeziku *VEE Pro* prikazane su u Tabeli 2.1.

Tabela 2.1 Skraćenice u programu VEE Pro

Skraćenica	**Opis radnje**
	Kontrola dokumenata
CTRL+N	Potpuno brisanje svih objekata, koji se nalaze na radnoj površini za pisanje programa. Potpuno isti efekat se dobije kao da se u glavnom meniju izabrala opcija *File => New*. Da se ne bi izgubio predhodno aktivan program, pojaviće se dijalog prozor za spašavanje stare verzije programa.
CTRL+O	Otvaranje već postojećeg *VEE Pro* dokumenta. Potpuno isti efekat se dobije kao da se u glavnom meniju izabrala opcija *File => Open.*
CTRL+S	Spašavanje trenutno aktivnog *VEE Pro* dokumenta. Potpuno isti efekat se dobije kao da se u glavnom meniju izabrala opcija *File => Save.*
CTRL+W	Spašavanje trenutno aktivnog *VEE Pro* dokumenta u novi dokumenat.
Alt+Print Screen	Kopiranje trenutno vidljivog programa u druge programe. Potpuno isti efekat se dobije kao da se u glavnom meniju izabrala opcija *File => Print Scrin.*
CTRL+E	Potpuno zatvaranje trenutno otvorenog dokumenta. Potpuno isti efekat se dobije kao da se u glavnom meniju izabrala opcija *File => Exit.*
	Izmjene u programu
Alt+...	Pristup glavnom meniju programa. *F-File, E-Edit, V-View*, *B-Bebug, L-Flow, D-Device, I-I/O, T-Data, S-Display, W-Windows, H-Help.*
CTRL+A	Selektovanje svih objekata u trenutno aktivnom programu.

Skraćenica	**Opis radnje**
CTRL+Lijevi taster miša	Selektovanje jednog po jednog objekta klikom miša. Selektovanje više objekata držanjem dugmeta CTRL i lijevog tastera miša, a zatim pomjeranjem miša na objekte, koji se žele selektovati.
Desni klik miša na prazan ekran	Otvara se padajući meni sa svim osobinama koje se mogu podesiti u programu.
Desni klik miša na objekat	Otvara se padajući meni sa svim osobinama koje se mogu podesiti u tom objektu.
	Izmjena u objektu formula
Enter	Prelazak u novi red za pisanje formule.
Delete	Brisanje teksta udesno.
Backspace	Brisanje teksta ulijevo.
streliceGore/Dole Lijevo/Desno	Kretanje kroz tekst.
Shift+End	Skok na zadnje slovo prvog reda.
Shift+Home	Skok na prvi karakter prvog reda.
	Izmjena u kodu nekog objekta
CTRL+C	Kopiranje transakcije.
CTRL+X	Isijecanje transakcije.
CTRL+V	Past transakcije (postavljanje iskopiranog dijela).
CTRL+Z	Undo (korak unazad) zadnje transakcije.
CTRL+Y	Redo (korak unaprijed) zadnje transakcije.
CTRL+Up	Pomjeranje trenutne transakcije naviše.
CTRL+Down	Pomjeranje trenutne transakcije nadole.
Shift+Up/Down	Selektovanje više transakcija istovremeno.
Esc	Izlazak iz objekta u kome se kuca kod.
	Rad sa objektima
CTRL+C	Kopiranje objekta. Potpuno isti efekat se dobije kao da se u glavnom meniju izabrala opcija *File => Copy.*
CTRL+L držati lijevi taster miša i pomjeriti miša	Kopiranje objekta na mjesto gdje smo pomjerili miša.
CTRL+X	Isijecanje objekta iz programa. Potpuno isti efekat se dobije kao da se u glavnom meniju izabrala opcija *File => Cut.*
CTRL+V	Postavljanje iskopiranog objekta na radnu površinu za pisanje programa. Potpuno isti efekat se dobije kao da se u glavnom meniju izabrala opcija *File => Paste.*
CTRL+A	Dodavanje novog ulaza (odnosno izlaza) za signale na objekat. Potrebno je prije toga da se klikne mišem na ulaz (ili izlaz), koji već postoji na objektu.

Skraćenica	**Opis radnje**
CTRL+D	Brisanje postojećeg ulaza (odnosno izlaza) za signale na objekat. Potrebno je prije toga da se klikne mišem na ulaz (ili izlaz), koji već postoji na objektu.
Shift+CTRL+ lijevi taster miša	Držati tipke Shift+CTRL i kada se dođe do neke linije pojave se crvene makazice umjesto kursora miša. Klikom na lijevi taster miša briše se ta linija. Može se koristiti komanda Undo za vraćanje obrisane linije u verziji *VEE Pro* 8.5 i novijoj.
	Navigacija kroz program
CTRL+M	Skok iz koda podprograma ili funkcije u kojoj se trenutno nalazimo, u glavni program.
CTRL+Tab ili CTRL+F6	Kretanje ciklično kroz kod svih procedura i funkcija koje postoje u programu.
Page Down/Up	Kretanje kroz radnu površinu programa za veličinu jednog ekrana dole/gore.
Shift+strelica ←↑↓→	Kretanje kroz radnu površinu programa lijevo/gore/dole/desno.
Shift+Page Down	Kretanje kroz radnu površinu programa za veličinu jednog ekrana lijevo.
Shift+Page Up	Kretanje kroz radnu površinu programa za veličinu jednog ekrana desno.
Home	Pomjeranje radne površine koda programa, tako da bude vidljiv gornji lijevi objekat.
Shift+ Home	Pomjeranje radne površine koda programa, tako da bude vidljiv donji desni objekat.
CTRL +F	Otvaranje prozora *Find* za pronalaženje željenog objekta.
Tab	Kretanje kroz menije, koji se pojavljuju na ekranu.
	Debaging programa
CTRL+G ili F5	Pokretanje programa. Potpuno isti efekat se dobije kao da se u glavnom meniju izabrala opcija *Debug => Run/Resume.*
Shift + F5	Zaustavljanje izvršenja programa, ali samo u verziji programa *VEE Pro* 8.5 i novijoj. Potpuno isti efekat se dobije kao da se u glavnom meniju izabrala opcija *Debug => Stop.*
CTRL+P	Pauza u izvršenju programa. Potpuno isti efekat se dobije kao da se u glavnom meniju izabrala opcija *Debug => Pause.*
CTRL+T ili F11	Izvršenje programa korak po korak, od jednog do drugog objekta. Potpuno isti efekat se dobije kao da se u glavnom meniju izabrala opcija *Debug => Step InTo.*
F10	Izvršenje programa korak po korak, od jednog do drugog objekta. Potpuno isti efekat se dobije kao da se u glavnom meniju izabrala opcija *Debug => Step Over.*

Skraćenica	Opis radnje
Shift +F11	Izvršenje programa korak po korak, od jednog do drugog objekta. Potpuno isti efekat se dobije kao da se u glavnom meniju izabrala opcija *Debug => Step Out.*
Shift + lijevi taster miša na liniji	Da se vide vrijednosti koje su bile na toj liniji prilikom zadnjeg pokretanja programa. Ovo naročito ima primjenu kod objekata tipa niza (*Array*) ili zapisa (*Record*). Potpuno isti efekat se dobije kao da se u glavnom meniju izabrala opcija *Debug => Line Probe.*
CTRL+B	Pravljenje kontrolnih tačaka, nad objektom ili grupom objekata, koji su predhodno markirani. Kada se program pokrene on će se izvršavati samo do ovih kontrolnih tačaka. Potpuno isti efekat se dobije kao da se u glavnom meniju izabrala opcija *Debug => Toggle Breakpoint.*
F1	Kada je objekat u programu selektovan, a zatim se pritisne tipka F1 otvoriće se prozor sa tekstom pomoći za taj objekat.
CTRL+U	Da nazivi ulaznih/izlaznih priključaka na objektu budu vidljivi ili nevidljivi.

2.5 PRAVLJENJE PRVOG PROGRAMA

U ovom poglavlju opisani su osnovni koraci za kreiranje programa, kroz izradu jednostavnog primjera. Kroz ovaj primjer početnici u radu sa ovim programskim jezikom mogu da uoče veliku jednostavnost i lakoću pravljenja programa. Programski jezik *VEE Pro* se najlakše uči kroz rješavanje praktičnih problema. Zadatak opisanog primjera je pravljenje laboratorijske vježbe za mjerenje jednosmjernog napona. Potreban je naponski izvor, koji generiše napon do 12 V. Izlaz sa naponskog izvora potrebno je mjeriti sa digitalnim voltmetrom od pet digita, analognim vlotmetrom, koji ima linearnu skalu do 10 V i analognim voltmetrom koji ima logoritamsku skalu do 10 V. Potrebno je postaviti vizuelnu i zvučnu signalizaciju, kada na ulaz analognih voltmetara dođe napon veći od 10 V.

Osnovni koraci za kreiranje programa u programskom jeziku *VEE Pro* su:

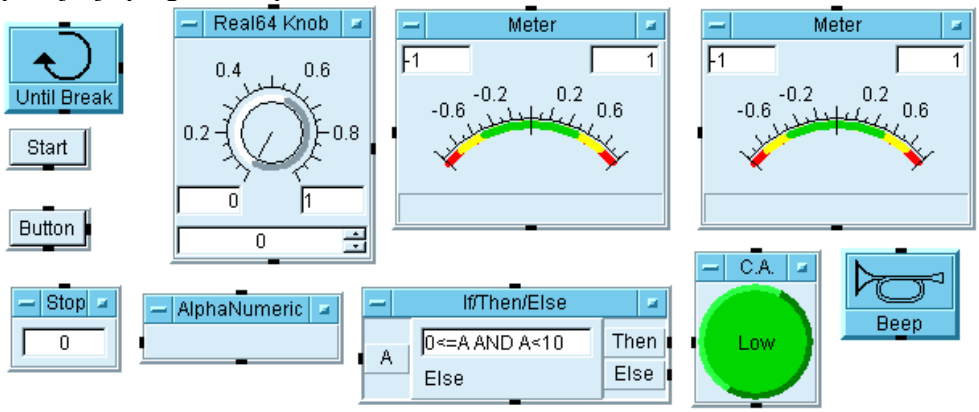
1) Izbor i postavljanje na radnu površinu potrebnih objekata za rješavanje postavljenog zadatka.
2) Definisanje osobina svakog postavljenog objekta.
3) Međusobno povezivanje objekata.
4) Pravljenje korisničkog interfejsa koji će biti vidljiv za korisnika, kada se program pokrene (*Add To Panel*).
5) Testiranje programa.
6) Pravljenje izvršne verzije programa (*Run Time Verion*).

2.5.1 Izbor potrebnih objekata za rješavanje postavljenog zadatka

Prva i najbitnija faza u procesu pisanja programa je definisanje problema. Preskakanje ove faze znači sigurno velike probleme u svim narednim fazama koje slijede. U ovoj fazi treba da se uoči problem, određuje se način rješavanja, vrši se analiza postojećeg stanja, analiziraju se iskustva u radu sa ovakvim i sličnim zadacima. Kada je izvršena postavka zadatka, može se pristupiti pravljenju algoritma. Zatim je potrebno izvršiti izbor objekata, koji su nam potrebni za rješavanje postavljenog zadatka. Za pisanje programa u programskom jeziku *VEE Pro*, sa kojim želimo da riješimo postavljeni problem, potrebni su nam sljedeći objekti:

- objekat za beskonačno ponavljanje programa (*Until Break*),
- komandno dugme za početak i kraj programa (*Start, Button*),
- objekat za kraj programa (*Stop*),
- virtuelni generator realnih brojeva u obliku kružnog potenciometra (*Real64 Knob*),
- dva indikatora za simulaciju analognih instrumenata (*Meter*),
- indikator za prikaz brojne vrijednosti (*Alpha Numjeric*),
- indikator za prikaz alarma u boji (*Color Alarm*),
- dva objekta uslovnog grananja (*If/Then/Else*),
- objekat za generisanje zvučnog signala (*Beep*).

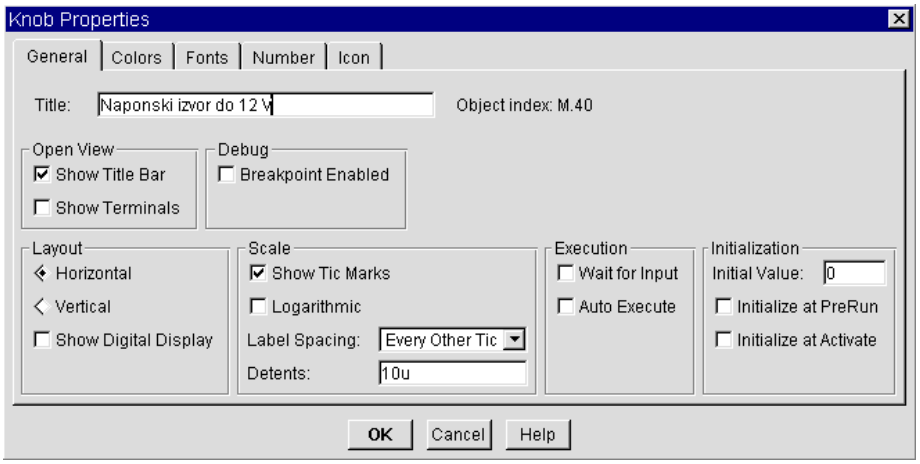
Svi ovi objekti se biraju iz palete *Meni* i postavljaju na radnu površinu za pravljenje programa, a prikazani su na slici 2.4.



Slika 2.4 Postavljanje potrebnih objekata na radnu površinu

2.5.2 Definisanje osobina objekata

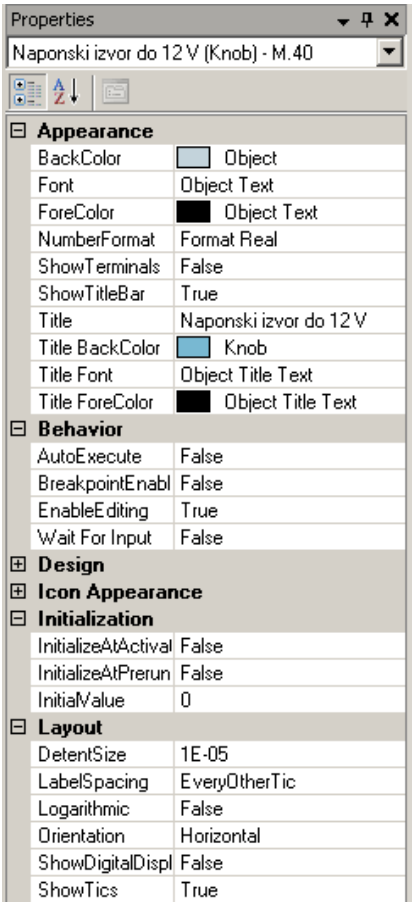
Kada se objekti postave na radnu površinu za pravljenje programa potrebno je izvršiti podešavanje osobina svakog objekta. Na taj način se objekti prilagođavaju njihovoj namjeni. U verzijama programa do *VEE Pro* 6.0 podešavanje osobina se



Slika 2.5 Prozor Properties u veriji VEE Pro 6.0

vrši u prozoru *Propertes*, do koga se dolazi duplim klikom miša na objekat, koji se podešava ili desni klik na objekat nakon čega se izabere opcija *Propertes*. Pojavljuje se prozor prikazan na slici 2.5. U novijim verzijama programa od *VEE Pro* 8.5 podešavanje osobina se vrši preko palete *Propertes*, koja se pojavljuje čim se markira željeni objekat. Pojavljuje se prozor prikazan na slici 2.6. Svaki objekat u programskom jeziku *VEE Pro* ima različite osobine, pa samim tim i različite palete za podešavanje. Preko ovih paleta se najčešće podešavaju neke od sljedećih osobina:

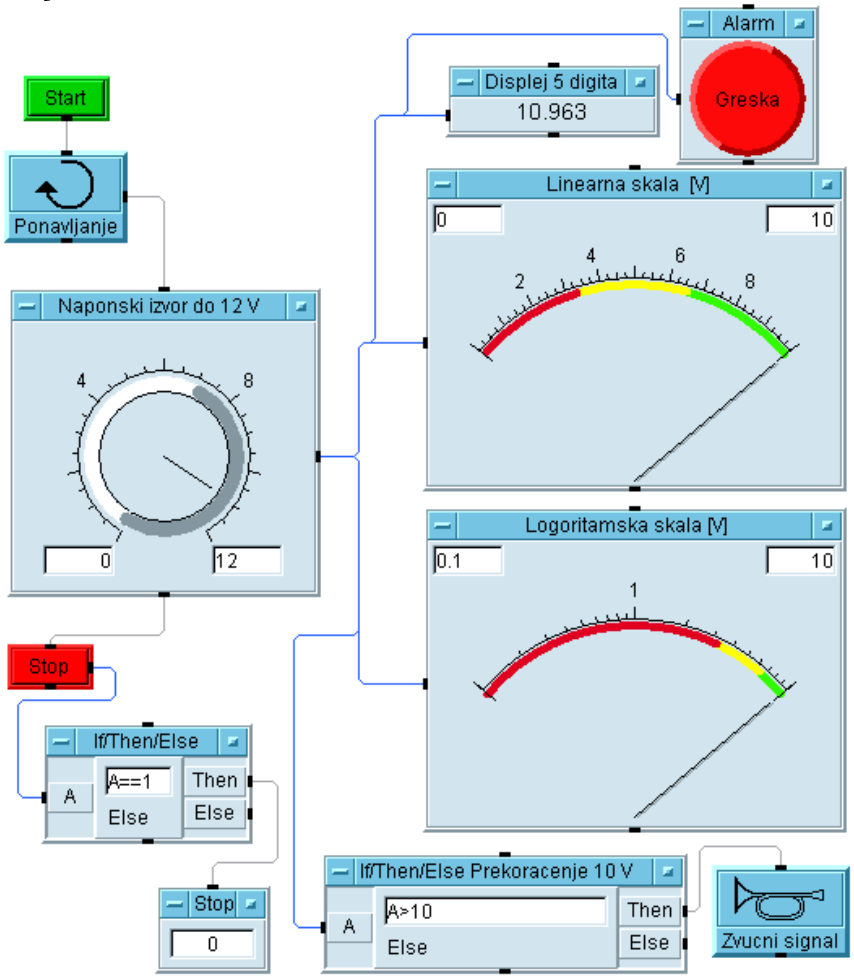
- natpis na objektu,
- ikonica na objektu,
- font, boja i veličina slova,
- boje pojedinih dijelova objekta,
- vidljivost ili nevidljivost pojedinih dijelova objekta,
- format prikaza brojnih vrijednosti,
- prikaz ulaza i izlaza iz objekta,
- postavljanje kontrolnih tačaka,
- način aktiviranja.



Slika 2.6 Verzija VEE Pro 8.5

2.5.3 Međusobno povezivanje objekata

Kada su objekti postavljeni na radnu površinu i kada su im postavljene željene osobine, potrebno je izvršiti povezivanje svih objekata. Objekti se međusobno povezuju linijama, povlačenjem miša od izlaza jednog objekta do ulaza drugog objekta. Pri tome se lijevi taster miša drži pritisnut. Izgled povezanih objekata prikazan je na slici 2.7.



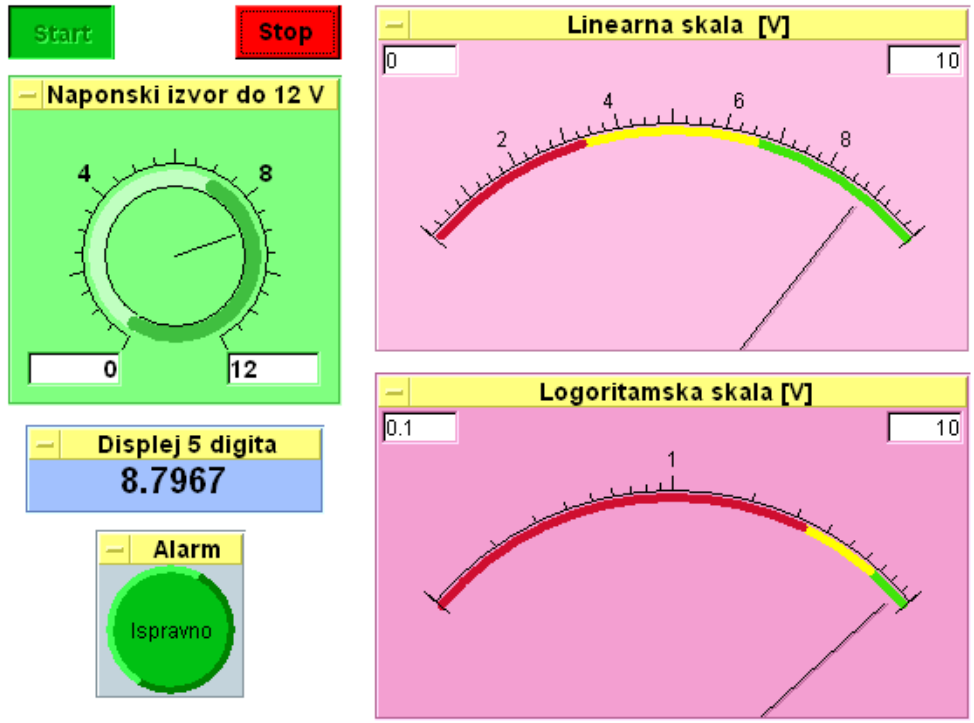
Slika 2.7 Izgled povezanih objekata

Ulazi za prihvat vrijednosti u objekat se uvijek nalaze sa lijeve, a izlazi sa desne strane objekta. Ako je na objektu čekirana opcija *Show Terminal* tada se vidi i naziv svakog ulaza/izlaza, ali ako ova opcija nije čekirana, tada se ulazi/izlazi prepoznaju samo po crnim pravougaonicima, koji se nalaze na ivici objekta. Ulaz

koji se nalazi na vrhu objekta služi samo za aktiviranje objekta, ali ne i za upis nekih vrijednosti u objekat. Izlaz, koji se nalazi na dnu objekta se koristi samo za aktiviranje jednog ili više drugih objekata, koji će se izvršavati u narednom programskom ciklusu. Na taj način se određuje redoslijed izvršavanaja programa. Svi ulazi u objekte (koji se nalaze sa lijeve strane) moraju biti povezani sa izlazom nekog objekta, koji generiše odgovarajuću vrijednost. Ako čak i jedan ulaz u objekat nije spojen neće biti moguće pokrenuti programa i pojaviće se poruka greške. Međutim, svi izlazi iz objekta ne moraju biti spojeni na druge objekte da bi program ispravno radio.

2.5.4 Pravljenje korisničkog interfejsa

Kada se izvrši povezivanje svih objekata, koji čine program dobije se korisnički interfejs na kome postoje objekti koji nisu bitni za običnog korisnika dok koristi program. Takođe se pojavljuju linije koje međusobno povezuju objekte i one samo mogu da zbune korisnika. Ovaj korisnički interfejs nosi naziv *To Detail*. Zbog toga je potrebno da se napravi korisnički interfejs, koji će biti puno pregledniji, jasniji i estetski dotjeran za potrebe korisnika. Izgled jednog takvog korisničkog interfejsa prikazan je na slici 2.8.



Slika 2.8 Korisnički interfejs

Korisnički interfejs se dobije tako što se na radnoj površini *To Detail* markiraju samo oni objekti koje želimo da korisnik vidi, a zatim se u paleti *Edit* izabere opcija *Add To Panel*. Nakon ove komande svi markirani objekti se pojavljuju u novoj radnoj površini, koja se zove *To Panel* i u njoj se vide samo objekti, a ne i linije koje ih međusobno povezuju. U *To Panel* radnoj površini se može dodatno pomjerati i podešavati svaki objekat. Programer u svakom trenutku može prelaziti iz jedne u drugu radnu površinu i mijenjati program.

2.5.5 Testiranje programa i pravljenje izvršne verzije programa

Kada se program napiše, potrebno je prvo uraditi njegovo testiranje. Prilikom testiranja programa otkrivaju se greške u pisanju programa. Greške mogu da budu sintaktičke i logičke. Sintaktičke greške nastaju prilikom kucanja naredbi u programu. U programskom jeziku *VEE Pro* to su naredbe koje se kucaju u gotovim objektima od kojih se sastavlja program. Najčešće greške su izostavljanje nekog slova ili da se neko slovo otkuca dva puta. Ove greške se puno lakše otkrivaju od logičkih grešaka. Logičke greške nastaju zbog pogrešnog rasporeda objekata u programu, tako da program ne može da se završi do kraja ili se na kraju dobije rezultat, koji nije dobar. Jedan od načina otklanjanja logičkih grešaka je da se u programu postave kontrolne tačke. Zatim da se program izvršava postepeno od jedne do druge kontrolne tačke, pri čemu se provjeravaju vrijednosti promjenljivih i izlazi svakog objekta. Drugi način otklanjanja logičkih grešaka je da se program izvršava korak po korak od jednog do drugog objekta, korišćenjem opcije *Step Into*. Na ovaj način se i najlakše dolazi do otkrivanja grešaka, a zatim i ispravljanja grešaka u programu.

Kada se program u potpunosti istestira, potrebno je napraviti izvršnu verziju programa preko opcije *Create Run Time Version* u glavnom meniju *File*. Na ovaj način se od razvojne verzije programa koja ima ekstenziju "*.VEE*", pravi izvršna verzija programa koja ima ekstenziju "*.VXE*". Korisnik bi trebao da radi samo sa izvršnim verzijama programa. Ako korisnik radi sa razvojnom verzijom programa, tada vrlo lako može da pokvari program pomjeranjem ili brisanjem samo jedne linije za povezivanje objekata. Sa druge strane pisac programa štiti svoj autorski rad isporučivanjem korisniku samo izvršne verzije programa.

3 TIPOVI PODATAKA

Sve komande u računaru se izvršavaju u mašinskom jeziku. Mašinski jezik radi samo sa podacima predstavljenim u obliku nizova binarnog alfabeta, odnosno sa nulama i jedinicama. Međutim, za običnog čovjeka binarni zapis je teško razumljiv i težak za izvođenje čak i osnovnih matematičkih operacija. Ljudi su naučeni da rade sa slovima i decimalnim brojevima, sa kojima većina lako izvode razne matematičke operacije. Zato su i nastali viši programski jezici gdje se naredbe za komunikaciju sa računarem pišu preko slova i brojeva. Da bi se olakšalo programiranje i kontrolisala upotreba memorije računara uvedeni su tipovi podataka. Tip podataka je skup vrijednosti koje imaju izvjesne zajedničke karakteristike. Najznačajnija od ovih karakteristika je skup operacija koje su definisane nad vrijednostima tog tipa. Za svaki tip podataka su definisane određene operacije, ali i gotove funkcije u svakom programskom jeziku. Kapacitet RAM memorije računara je uvijek bio jedan od glavnih ograničavajućih faktora za brzinu rada računara. Različiti tipovi podataka zauzimaju i različit kapacitet memorije, pa zato pravilan izbor tipova podataka u programu može značajno da ubrza ili uspori rad računara. Ovo posebno dolazi do izražaja kod pisanja programa, koji imaju veliki broj promjenljivih.

Osnovni tipovi i strukture podataka se mogu se podijeliti na:
- statički skalarni tipovi (elementarni podaci koji su skalari),
- statički struktuirani tipovi (povezuju elementarne podatke u preciznu strukturu),
- dinamički tipovi sa promjenljivom veličinom,
- dinamički tipovi sa promjenljivom strukturom.

Pod statičkim tipovima[6] (ili tipovima sa statičkom strukturom) podrazumijevamo tipove podataka kod kojih je unaprijed fiksno definisana unutrašnja struktura svakog podatka, a veličina (koliko memorije zauzima) fiksno se definiše prije (ili u vrijeme) izvršenja programa, koji koristi podatke statičkog tipa. Statički tipovi podataka obuhvataju skalarne i struktuirane podatke.

Pod skalarnim tipovima podrazumijevamo najprostije tipove podataka čije su vrijednosti skalari, odnosno takve veličine koje se tretiraju kao elementarne cjeline i za koje nema potrebe da se dalje razlažu na prostije komponente.

Pod struktuiranim tipovima podataka podrazumijevamo sve složene tipove podataka, koji se realizuju povezivanjem nekih elementarnih podataka u precizno

[6] Dr Jozo J. Dujmović, *Programski jezici i metode programiranja*, Akademska misao, Beograd, 2003, str 2.3.

definisanu strukturu. U ulozi elementarnih podataka obično se pojavljuju skalari ili neke jednostavnije strukture. Primjer struktuiranog tipa podataka je zapis (eng. *Record*). Na primjer zapis **student** može da se sastoji od polja: **broj indeksa** (tip *Int64*), **ime i prezime** (tip *Text*), **datum rođenja** (tip *Date/Time*), **godina studija** (tip *UInt8*) i **smjer** (tip *Text*). Skalarni tipovi podataka mogu biti linearno uređeni ili linearno neuređeni. Linearno uređeni tipovi podataka su tipovi kod kojih se vrijednosti osnovnog skupa preslikavaju na jedan interval iz niza cijelih brojeva. Tada se za svaki podatak zna redni broj podatka. Stoga svaki podatak izuzev početnog ima svog predhodnika u nizu, odnosno svaki podatak ima svog slijedbenika izuzev krajnjeg.

Pod dinamičkim tipovima podataka podrazumijevamo tipove podataka kod kojih se veličina i/ili struktura podataka slobodno mijenja u toku obrade. Kod dinamičkih tipova sa promjenljivom veličinom podrazumijevamo da je struktura podataka fiksna, ali se njihova veličina dinamički mijenja tokom obrade. Saglasno tome se dinamički mijenjaju i memorijski zahtjevi. Primjer dinamičkog tipa podataka je niz (eng. Array), koji može da ima promjenljiv broj elemenata, ali svi njegovi elementi moraju biti istog tipa.

Kod dinamičkih tipova sa promjenljivom strukturom unaprijed je fiksno definisan jedino princip po kome se formira struktura podataka, dok se sama konkretna struktura i količina podataka u memoriji slobodno dinamički mijenjaju. U programskom jeziku *VEE Pro* nema primjera dinamičkih tipova podataka sa promjenljivom strukturom.

Sve promjenljive koje će se koristiti u programu, prije njihove upotrebe moraju se eksplicitno definisati i specificirati tip, osim za univerzalni tip podataka variant.

3.1 LOGIČKE PROMJENJIVE (*BOOLEAN*)

Logičke promjenjive (eng. *Boolean*) predstavljaju najjednostavnije promjenjive. One zauzimaju i najmanje memorijskog prostora prilikom pokretanja programa. Standarni identifikatori *true* i *false* označavaju dvije moguće logičke vrijednosti: istina (1) i laž (0). Odnosno u elektrotehnici to su dva stanja uključeno (*On*) ili isključeno (*Off*). Primjenom relacionih operatora =, >, <, >=, <=, <> dobijaju se veličine logičkog tipa. U programskom jeziku *VEE Pro* operator = se koristi za dodjelu vrijednosti, a operator za poređenje dvije promjenljive je ==.

Osnovne logičke operacije nad logičkim tipom podataka su :

1) *NOT* - negacija
2) *AND* - konjukcija (i)
3) *OR* - disjunkcija (ili)

Negacijom se dobija suprotna logička vrijednost i za nju vrijedi

not(false) = true
not(true) = false.

Konjukcijom dva logička izraza se dobije tačna vrijednos, samo ako obadva (ili svi izrazi ako ih ima više) izraza za rezultat imaju tačnu vrijednost i za nju vrijedi

false and false = false
false and true = false
true and false = false
true and true = true

Disjunkcijom dva logička izraza se dobije tačna vrijednos, ako bilo koji od dva (ili samo jedan izrazi ako ih ima više) izraza za rezultat imaju tačnu vrijednost i za nju vrijedi

false or false = false
false or true = true
true or false = true
true or true = true

Izrazi u kojima se primjenjuju logičke promjenljive nazivaju se logički izrazi. Primjer nekih ispravno napisanih logičkih izraza u objektu grananja *If/Then/Else*

(A==B AND A>5) OR B<1000
(A>B OR A>5) AND B<1000
(A==B OR A>5) AND (A<100 OR B<1000).

3.2 CIJELI BROJEVI (*INTEGER*)

Cjelobrojni tip podataka (eng. *Integer*) predstavlja najjednostavniji brojni tip podataka. On predstavlja podskup skupa cijelih brojeva. Koji je to tačno podskup, zavisi od konkretne definicije ovog tipa u programskom jeziku *VEE Pro*. Definicija sadrži broj koji govori sa koliko bita se predstavlja taj decimalni broj u binarnom obliku, što automatski određuje minimalni (početni) i maksimalni (krajnji) broj koji ovaj tip podatka može da uzme. U programskom jeziku *VEE Pro* postoje sljedeće definicije cjelobrojnog tipa podataka:

1) **UInt8** je 8-bitno predstavljanje, a ovaj tip podataka sadrži samo pozitivne cjelobrojne brojeve u rasponu (0 do 255).
2) **UInt16** je 16-bitno predstavljanje, a ovaj tip podataka sadrži samo pozitivne cjelobrojne brojeve u rasponu (0 do 65535).
3) **Int16** je 16-bitno predstavljanje, a ovaj tip podataka sadrži pozitivne, ali i negativne cjelobrojne brojeve u rasponu (-32768 do 32767).
4) **Int32** je 32-bitno predstavljanje, a ovaj tip podataka sadrži pozitivne, ali i negativne cjelobrojne brojeve u rasponu (-2147483648 do 2147483647).
5) **Int64** je 64-bitno predstavljanje, a ovaj tip podataka sadrži pozitivne, ali i negativne cjelobrojne brojeve u rasponu (-9223372036854775808 do 9223372036854775807).

Kada se piše program za svaki objekat i promjenljivu, a na osnovu maksimalno očekivanog broja koji se može pojaviti, treba izabrati odgovarajući tip cijelog

broja. Tip se bira na osovu predstavljenih raspona brojeva, kako se memorija ne bi nepotrebno trošila i time usporavalo izvršenje programa.

Kao i na svakom drugom standardnom tipu, tako je i na tipu *Integer* definisan tačno određeni skup operacija. To su u ovom slučaju uobičajene cjelobrojne aritmetričke operacije: sabiranje (+), oduzimanje (-), množenje (*), cjelobrojno dijeljenje (DIV) i ostatak cjelobrojnog dijeljenja (MOD).

Primjer:

3.2 DIV 1 = 3	3.2 MOD 1 = 0.2
3.2 DIV 2 = 1	3.2 MOD 2 = 1.2
3.2 DIV 3 = 1	3.2 MOD 3 = 0.2
3.2 DIV 4 = 0	

U programskom jeziku *VEE Pro* postoji dosta gotovih funkcija koje rade sa ovim tipom podataka.

3.3 REALNI BROJEVI (*REAL*)

Realni tip podataka (eng. *Real*) predstavlja podskup skupa realnih brojeva. Koji je to tačno podskup, zavisi od konkretne definicije ovog tipa u programskom jeziku *VEE Pro*. Definicija sadrži broj koji govori sa koliko bita se predstavlja taj decimalni broj u binarnom obliku, što automatski određuje minimalni (početni) i maksimalni (krajnji) broj koji ovaj tip podatka može da uzme. Za ovaj tip podataka je bitno da se uvijek navede broj značajnih cifri. Značajna cifra predstavlja broj, koji pokazuje koliko će realni broj imati prikazanih cifri iza decimalne tačke. Ovo je potrebno uraditi kako nam program ne bi automatski dodijelio 4, 8 ili 16 značajnih cifri iza decimalne tačke realnog broja. U programskom jeziku *VEE Pro* postoje sljedeće definicije realnog tipa podataka:

1) **Real32** je 32-bitno predstavljanje, a ovaj tip podataka sadrži realne brojeve u rasponu (±3.40282347E±38, sa približno 8 značajnih decimalnih mjesta).
2) **Real64** je 64-bitno predstavljanje, a ovaj tip podataka sadrži realne brojeve u rasponu (±1.7976931348623157E308, sa približno 16 značajnih decimalnih mjesta).

Na tipu *Real* definisane su osnovne aritmetričke operacije: sabiranje (+), oduzimanje (-), množenje (*) i dijeljenje (/). U programskom jeziku *VEE Pro* postoji dosta gotovih funkcija koje rade sa ovim tipom podataka. Neke od funkcija koje se najčešće koriste za konverziju realnog u cijeli broj su:

1) ***round*** - zaokruživanje realnog broja na bližu cjelobrojnu vrijednost.
2) ***intPart*** - uzimanje samo cjelobrojne vrijednosti realnog broja.
3) ***fracPart*** - uzimanje realnog dijela realnog broja.
4) ***floor*** - zaokruživanje realnog broja na prvu manju cjelobrojnu vrijednost.

Primjer:
round([23.0, 23.1, 23.9, 23.5, (-23.5), 24.5]), daje [23, 23, 24, 24, -23, 24].
intPart([23.0, 23.1, 23.9, 23.5, (-23.5), 24.5]), daje [23, 23, 23, 23, -23, 24].
fracPart([23.0, 23.1, 23.9, 23.5, (-23.5), 24.5]), daje [0, 0.1, 0.9, 0.5, -0.5, 0,5].
floor([23.0, 23.1, 23.9, 23.5, (-23.5), 24.5]), daje [23, 23, 23, 23, -24, 24].

3.4 TEKST (*TEXT*)

Znakovni tip podataka (eng. *Text*) predstavlja skup: malih i velikih slova engleskog alfabeta, cifri od 0 do 9, specijalnih karaktera i kontrolnih znakova. Ovaj tip podataka je značajan, jer kada korisnik komunicira sa računarom preko ulaznih i izlaznih uređaja, vrlo je bitno da se podaci pojavljuju u formi koja je čitljiva za čovjeka. Da bi se tekst razlikovao od naziva promjenljivih, ulaza ili izlaza iz objekata potrebno je da stoji između apostrofa ("tekst koji se navodi"). Kada se posmatra izlaz iz nekog objekta (na kontrolnoj liniji koja spaja dva objekta) koji je znakovnog tipa, dobijeni tekst će uvijek biti prikazan između apostrofa.

ASCII kod (*Amjerican Standard Code for Information Interchange*[7]) je jedan od najznačanijih načina predstavljanja znakovnog tipa podataka. *ASCII* kod je poznat u 7-bitnoj verziji sa 128 znakova i 8-bitnoj verziji sa 256 znakova. *ASCII* kod sa 128 znakova prikazan je u tabeli 3.1.

U programskom jeziku *VEE Pro* postoji veliki broj gotovih funkcija koje rade sa ovim tipom podataka:

1) ***strDown(str)*** - vrši konverziju svih velikih u mala slova.
   [strDown("Mira%+#123"), daje "mira%+#123"]
2) ***strUp(str)*** - vrši konverziju svih malih u velika slova.
   [strDown("Mira%+#123"), daje "MIRA%+#123"]
3) ***strFromLen(str,from,len)*** - izdvaja pod tekst određene dužine, od startnog znaka.
   [strFromLen("Teodora je mala",0,6), daje "Teodor"]
   [strFromLen("Teodora je mala",9,2), daje "je"]
   [strFromLen("Teodora je mala",0,15), daje "Teodora je mala"]
4) ***strFromThru(str,from,thru)*** - izdvaja pod tekst, od startnog do zadnjeg znaka.
   [strFromLen("Teodora je mala",0,6), daje "Teodor"]
   [strFromLen("Teodora je mala",9,10), daje "je"]
5) ***strLen(str)*** - daje broj znakova u tekstu.
   [strLen("Mihailo"), daje 7] i [strLen("123456"), daje 6]
6) ***strRev(str)*** - daje tekst sa obrnutim redoslijeda znakova.
   [strRev("Mihailo") daje "oliahiM")]

[7] http://frank.harvard.edu/aoe/images/t10r3.pdf

Tabela 3.1 ASCII kod

non-printing					printing			printing			printing		
Name	Control char	Char	Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char	Hex	Dec
null	ctrl-@	NUL	00	00	SP	20	32	@	40	64	‘	60	96
start of heading	ctrl-A	SOH	01	01	!	21	33	A	41	65	a	61	97
start of text	ctrl-B	STX	02	02	"	22	34	B	42	66	b	62	98
end of text	ctrl-C	ETX	03	03	#	23	35	C	43	67	c	63	99
end of xmit	ctrl-D	EOT	04	04	$	24	36	D	44	68	d	64	100
enquiry	ctrl-E	ENQ	05	05	%	25	37	E	45	69	e	65	101
acknowledge	ctrl-F	ACK	06	06	&	26	38	F	46	70	f	66	102
bell	ctrl-G	BEL	07	07	’	27	39	G	47	71	g	67	103
backspace	ctrl-H	BS	08	08	(	28	40	H	48	72	h	68	104
horizontal tab	ctrl-I	HT	09	09	)	29	41	I	49	73	i	69	105
line feed	ctrl-J	LF	0A	10	*	2A	42	J	4A	74	j	6A	106
vertical tab	ctrl-K	VT	0B	11	+	2B	43	K	4B	75	k	6B	107
form feed	ctrl-L	FF	0C	12	,	2C	44	L	4C	76	l	6C	108
carriage return	ctrl-M	CR	0D	13	-	2D	45	M	4D	77	m	6D	109
shift out	ctrl-N	SO	0E	14	.	2E	46	N	4E	78	n	6E	110
shift in	ctrl-O	SI	0F	15	/	2F	47	O	4F	79	o	6F	111
data line escape	ctrl-P	DLE	10	16	0	30	48	P	50	80	p	70	112
device control 1	ctrl-Q	DC1	11	17	1	31	49	Q	51	81	q	71	113
device control 2	ctrl-R	DC2	12	18	2	32	50	R	52	82	r	72	114
device control 3	ctrl-S	DC3	13	19	3	33	51	S	53	83	s	73	115
device control 4	ctrl-T	DC4	14	20	4	34	52	T	54	84	t	74	116
neg acknowledge	ctrl-U	NAK	15	21	5	35	53	U	55	85	u	75	117
synchronous idle	ctrl-V	SYN	16	22	6	36	54	V	56	86	v	76	118
end of xmit block	ctrl-W	ETB	17	23	7	37	55	W	57	87	w	77	119
cancel	ctrl-X	CAN	18	24	8	38	56	X	58	88	x	78	120
end of medium	ctrl-Y	EM	19	25	9	39	57	Y	59	89	y	79	121
substitute	ctrl-Z	SUB	1A	26	:	3A	58	Z	5A	90	z	7A	122
escape	ctrl-[	ESC	1B	27	;	3B	59	[	5B	91	{	7B	123
file separator	ctrl-\	FS	1C	28	<	3C	60	\	5C	92	|	7C	124
group separator	ctrl-]	GS	1D	29	=	3D	61	]	5D	93	}	7D	125
record separator	ctrl-^	RS	1E	30	>	3E	62	^	5E	94	~	7E	126
unit separator	ctrl-_	US	1F	31	?	3F	63	_	5F	95	DEL	7F	127

7) ***strTrim(str)*** - pravi tekst bez praznih karaktera na početku i kraju teksta. [strTrim (" Teodora je mala "), daje "Teodora je mala"]
8) ***strTrim(str*, [znakovi koji se izbacuju])** - pravi tekst bez znakova navedenih iza zapete, pri čemu se izbacuju slova sa kraja početnog teksta. [strTrim("eeeABCdefg%+#","eg#"), daje "ABCdefg%+"]

9) ***intToChar*(broj)** - vrši konverziju broja u tekst, prema ASCII standardu.
[intToChar(65), daje slovo A] [intToChar(97), daje slovo a]
[intToChar(62), daje simbol >] [intToChar(37), daje simbol %]
10) ***charToInt*(slovo)** - vrši konverziju prvog slova u tekstu u broj, prema ASCII standardu.
[charToInt (A), daje broj 65] [charToInt (a), daje broj 97]
[charToInt (aprl), daje broj 97] [charToInt (Mirko), daje broj 77]

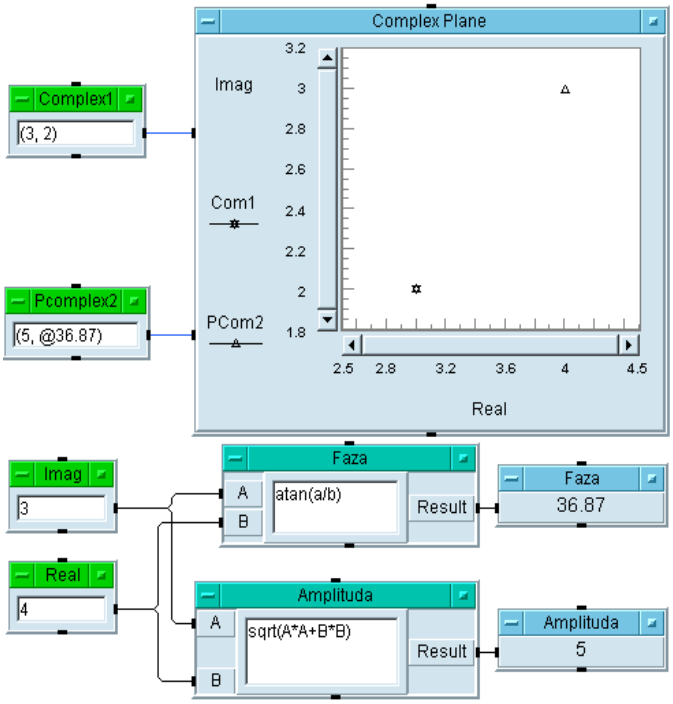
Poseban tekstualni tip podatka koji postoji u programu *VEE Pro* je *Enum*. To je tekst koji se pojavljuje kao izlaz iz objekata *Selection Control* (na primjer, *Radio Buttons* objekat). Ovaj tip podataka se ne koristi kao ulazni podatak u druge objekte, osim za displeje. Međutim, ovi objekti imaju i drugi izlaz (*Ordinal*) koji generiše cjelobrojne vrijednosti (*Integer*), koje se koriste kao ulaz u druge objekte.

3.5 KOMPLEKSNI BROJEVI (*COMPLEX*)

U programskom jeziku *VEE Pro* postoje dva kompleksna (eng. *Complex*) tipa podataka: *Complex* i *PComplex*.

Complex je tip podataka, koji služi za predstavljanje kompleksnih brojeva. Svaki kompleksni broj se predstavlja kao kombinacija realnog i imaginarnog dijela u obliku (realni dio, imaginarni dio). Svaka komponenta se zapisuje u formatu *Real64*. Na primjer kompleksni broj $\bar{A} = \mathbf{3} + \boldsymbol{i2}$ se predstavlja kao **(3,2)**.

PComplex je tip podataka, koji se takođe koristi za predstavljanje kompleksnih brojeva. Kod ovog tipa podataka kompleksni brojevi se predstavljaju kao kombinacija amplitude i faze u obliku (**amplituda, @faza**).

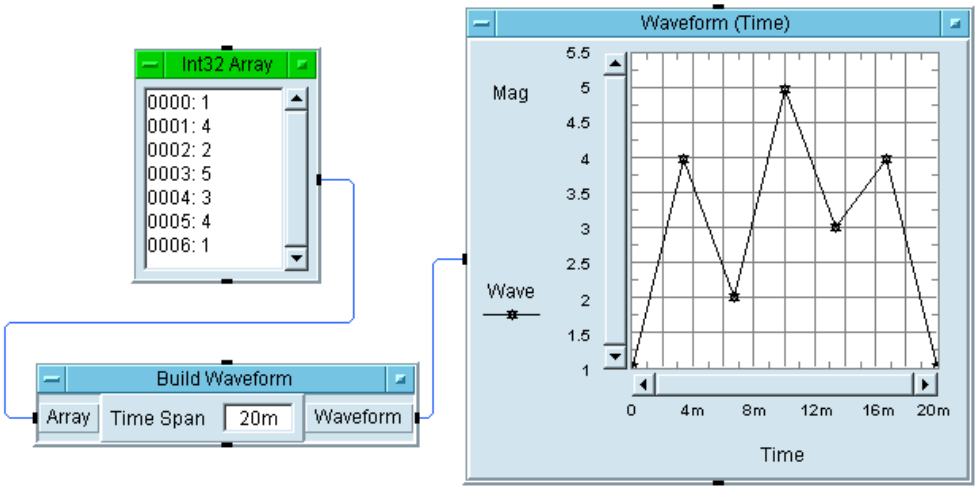


Slika 3.1 Kompleksni tipovi podataka

Faza kompleksnog broja se izražava u jedinici stepen. Svaka komponenta se zapisuje u formatu *Real64*. Na primjer kompleksni broj Ā = **4** + ***i*3** se predstavlja kao **(5, @36.87)**, a to je prikazano na slici 3.1.

3.6 TALASNI OBLIK (*WAVEFORM)*

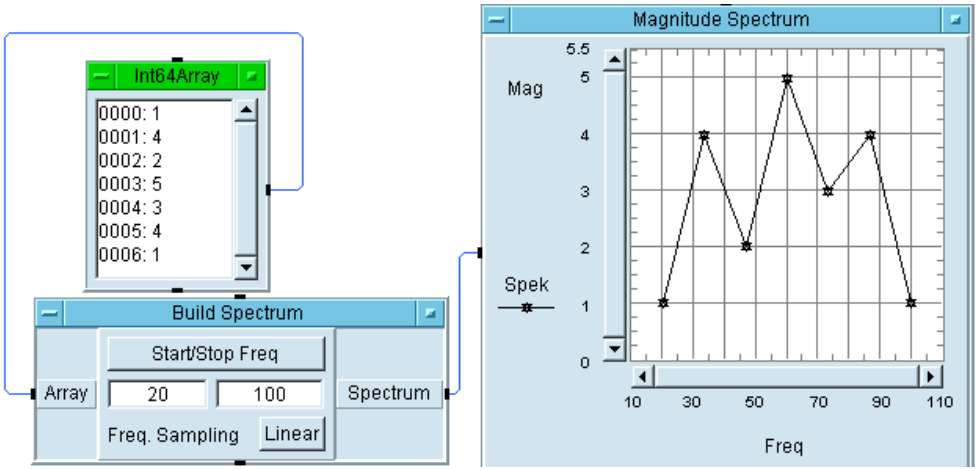
U programskom jeziku *VEE Pro* postoji tip talasni oblik (eng. *Waveform*), koji predstavlja podatke u vremenskom domenu. Talasni oblik sadrži brojne vrijednosti u formatu *Real64*. Ovaj tip podataka se formira pomoću objekta *Build Waveform*. Kao ulaz u ovaj objekat se koriste brojevi zapisani u obliku jednodimenzionog niza (*Array1*), a ove vrijednosti se na XY dijagramu prikazuju na Y osi. U ovom objektu se u polje *Time Span* unosi ukupni vremenski raspon između prve i zadnje vrijednosti ulaznog niza, a ove vrijednosti se na XY dijagramu prikazuju na X osi. Primjer generisanja ovog tipa podatka prikazan je na slici 3.2.



Slika 3.2 Waveform tip podataka

3.7 SPEKTAR (*SPECTRUM*)

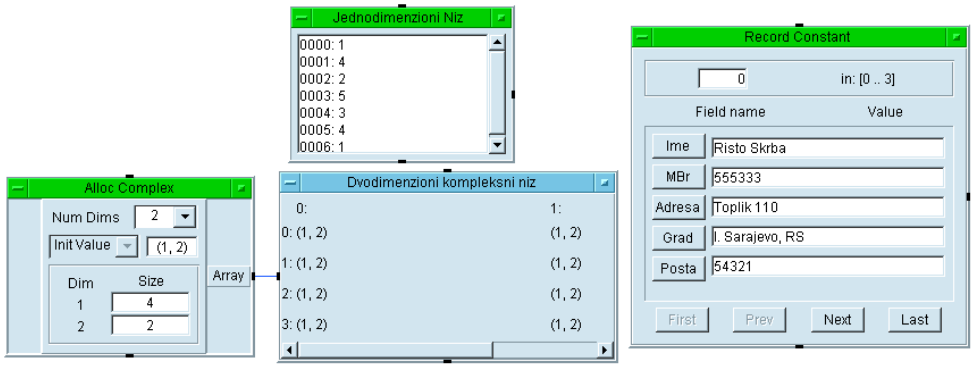
U programskom jeziku *VEE Pro* postoji spektar (eng. *Spectrum*) tip podataka u frekventnom domenu, koji sadrži vrijednosti u formatu *Real64*. Ovaj tip podataka se formira pomoću objekta *Build Spectrum*. Kao ulaz u ovaj objekat se koriste brojevi zapisani u obliku jednodimenzionog niza (*Array1*), a ove vrijednosti se na XY dijagramu prikazuju na Y osi. U ovom objektu se u polje *Start/Stop Freq* unosi frekvencijski raspon između prve i zadnje vrijednosti ulaznog niza, a ove vrijednosti se na XY dijagramu prikazuju na X osi. Razlika između dvije frekvencije na X osi može biti linearna ili logorotamska. Primjer generisanja ovog tipa podatka prikazan je na slici 3.3.



Slika 3.3 Spektar tip podataka

3.8 NIZ (*ARRAY*) I ZAPIS (*RECORD*)

Niz (eng. *Array*) i zapis (eng. *Record*) spadaju u složene tipove podataka. Niz je objekat, koji se sastoji od više skalarnih podataka (elemenata), ali koji su istog tipa. Elementi niza mogu biti svi do sada opisani tipovi podataka. Niz može biti jednodimenzioni i višedimenzioni. Jednodimenzioni niz se može predstaviti kao tabela, koja ima samo jednu kolonu i više redova. Dvodimenzioni niz se može predstaviti kao tabela, koja ima više kolona i više redova, ali su svi elementi niza istog tipa. Trodimenzioni niz se može predstaviti kao kocka, koja ima više kolona, više redova i više elemenata treće dimenzije (dubina kocke), ali su svi elementi niza istog tipa.



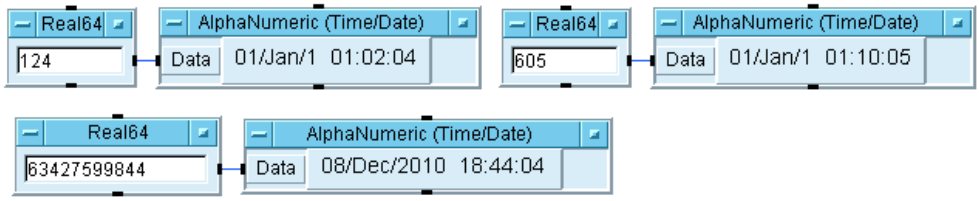
Slika 3.4 Tip podataka niz i zapis

Zapis (eng. *Record*) spada u složeni tip podataka, koji se najlakše može opisati kao tabela, koja ima više kolona i više redova. Svaka kolona mora da ima ime, koje mora biti različito od imena drugih kolona. Podaci u jednoj koloni moraju biti istog

tipa. Ali podaci u različitim kolonama mogu biti različitog tipa podataka. Red sadrži podatke iz svih kolona i u jednom redu mogu biti podaci različitog tipa. Najmanji element zapisa zove se polje. Jedno polje može imati samo jedan podatak. Podaci se u zapis unose red po red zapisa, ali tako da se unose polja za svaku kolonu. Primjer ovog tipa podataka u programskom jeziku *VEE Pro* prikazan je na slici 3.4.

3.9 DATUM I VRIJEME (*DATE/TIME*)

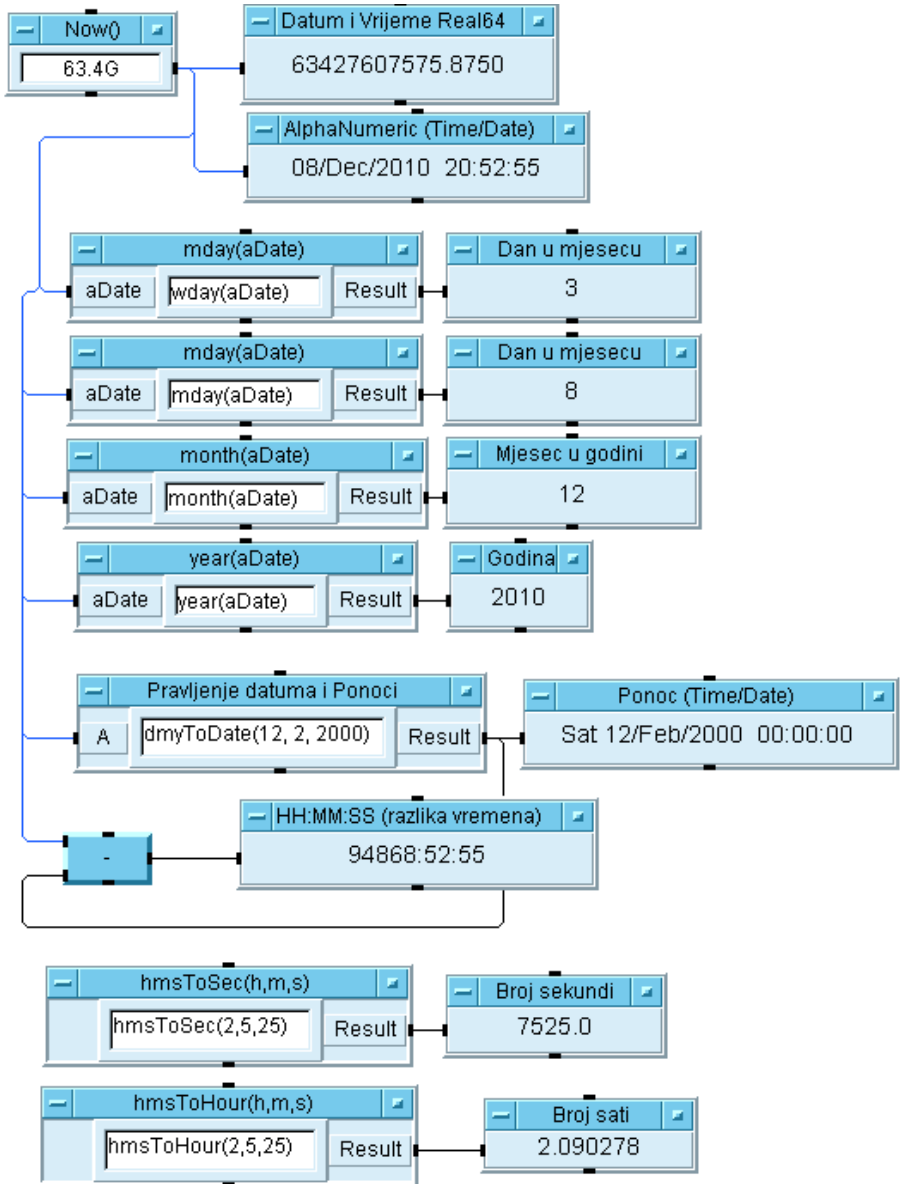
Datum i vrijeme (eng. *Date/Time*) tip podataka se koristi za prikaz datuma i vremena u raznim formatima. Ovaj tip podataka u suštini predstavlja realni broj u formatu *Real64*. Ovaj realni broj predstavlja broj sekundi od 1. januara 0001. godine poslije Hrista, prema *UTC* (*Universal Coordinated Time*) vremenu. *UTC* vrijeme predstavlja korekciju lokalne vremenske zone, koja je podešena na računaru (*Control Panel => Date and Time => Time Zone => (GMT +01:00) Belgrade, Ljubljana*), u odnosu na *GMT* (*Greenwich Mean Time*) vrijeme. U našoj zemlji *UTC* vrijeme iznosi +1 sat, jer naša vremenska zona prednjači u odnosu na *Greenwich* kod Londona za jedan sat. Na slici 3.5 data su tri primjera pretvaranja realnog broja u datum i vrijeme. Pretvaranjem broja 124 u datum i vrijeme dobije se: 01. januar 0001. godine 01:02:04, u našoj vremenskoj zoni. Da je na računaru bila podešena vremenska zona *Greenwich* +00, onda bi se dobio potpuno drugi rezultat za vrijeme i datum: 01. januar 0001. godine 00:02:04. Da bi alfanumjerički displej mogao da prikaže datum, potrebno je podesiti osobinu *Number*. Da bi mogli podešavati ovu osobinu opcija *Global Format* treba biti ne čekirana. Kod parametra *Real* se bira način prikaza brojeva, a ako se izabere opcija *Time Stamp* tada ovaj objekat prikazuje datum i vrijeme. Dodatno se može podesiti da se prikazuje samo datum, samo vrijeme ili datum i vrijeme zajedno.



Slika 3.5 Pretvaranje realnog broja u datum i vrijeme

Programski jezik *VEE Pro* ima dosta gotovih funkcija, koje omogućuju razne operacije sa datumima i vremenom, kao na slici 3.6. Neke od tih funkcija su:

1) ***now()*** - koja uzima sistemsko vrijeme sa računara, a na svom izlazu daje realni broj u formatu *Real64*. Dobijeni realni broj predstavlja broj sekundi od 1. januara 0001. godine poslije Hrista, pa do trenutka aktiviranja ove funkcije u programu.



Slika 3.6 Funkcije za rad sa datumom i vremenom

2) ***wday(aDate)*** - daje redni broj dana u sedmici, kada je ulazna vrijednost u ovu funkciju datum. Opseg vrijednosti izlaza ove funkcije je od 1 do 7. Primjer: wday(dmyToDate(28,1,2003)) daje 2, zato što je 28. januar 2003. godine bio utorak.

3) ***mday(aDate)*** - daje broj dana u trenutnom mjesecu, kada je ulazna vrijednost u ovu funkciju datum. Opseg vrijednosti izlaza ove funkcije je od 1 do 31.
4) ***month(aDate)*** - daje broj mjeseca u trenutnoj godini, kada je ulazna vrijednost u ovu funkciju datum. Opseg vrijednosti izlaza ove funkcije je od 1 do 12.
5) ***year(aDate)*** - daje broj godina u trenutnom datumu.
6) ***dmyToDate(d,m,y)*** - formira datum od 3 ulazna parametra. Parametar **d** je dan u mjesecu. Parametar **m** je mjesec u godini. Parametar **y** je godina.
7) ***hmsToSec(h,m,s)*** - daje ukupan broj sekundi od 3 ulazna parametra. Parametar **h** je broj sati. Parametar **m** je broj minuta. Parametar **s** je broj sekundi.

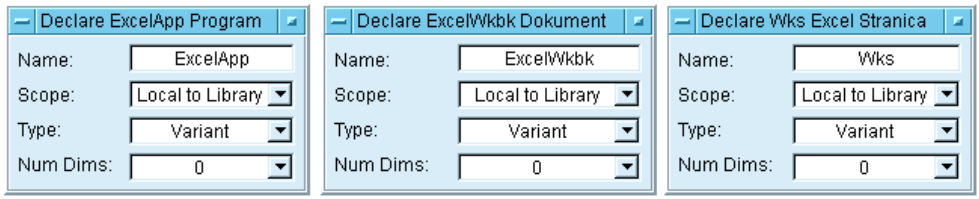
   Primjer: hmsToSec(2,5,25)= 2*3600 + 5*60 + 25 = 7525 sekundi.
8) ***hmsToHour(h,m,s)*** - daje ukupan broj sati od 3 ulazna parametra. Parametar **h** je broj sati. Parametar **m** je broj minuta. Parametar **s** je broj sekundi.

   Primjer: hmsToHour(2,5,25)= 2 + 5/60 + 25/3600 =2,090278 sati.

3.10 NEODREĐENI TIP (*VARIANT*)

Variant je neodređeni tip podataka, koji postoji u programskom jeziku *VEE Pro*, a vezan je za deklaraciju globalnih i lokalnih promjenljivih. Kada se neka promjenljiva deklariše kao varijant, to znači da ona može da poprimi bilo koji do sada opisani tip podatka. Promjenjva se prilagođava onom tipu podatka, koji dođe na njen ulaz. Međutim, ne treba pretjerivati sa korišćenjem ove promjenljive, pogotovo, kada smo sigurni da je promjenljiva tačno poznatog tipa podataka. Razlog leži u činjenici što ovaj tip promjenljive zauzima najviše memorije od svih ostalih tipova podataka.

Ovaj tip podatka se posebno koristi kod deklarisanja promjenljivih koje se koriste za povezivanje sa drugim programima instalisanim na računaru, a prvenstveno sa programima iz *Microsoft Office* paketa. U primjeru na slici 3.7 prikazano je kako se tip vatiant koristi za deklarisanje promjenljivih koje služe za povezivanje sa *Excel* programom, *Excel* dokumentom i *Excel* stranicom.



Slika 3.7 Variant Tip podataka

4 PALETE MENIJA

U ovom poglavlju predstavljena je paleta menija u programu *VEE Pro*. Biće detaljno predstavljeni svi elementi menija, kako bi se što lakše mogli koristiti za pisanje programa. Za pojedine elemente, koji predstavljaju objekte pomoću kojih se sastavlja program, biće dati i primjeri programa. Predstavljen je i način, kako da se pojedini objekti mogu kombinovati sa drugim objektima u jednu cjelinu, koja se zove program. Opisano je koji se objekti mogu iskoristili za rješavanje nekih praktičnih problema. Ponuđeni primjeri su djelimično uprošćeni, ali se nevelikim trudom mogu brzo i efikasno dovesti do profesionalne aplikacije. Obrađene oblasti su raznovrsne i brižljivo birane, kako bi čitaoci mogli da uvide višestruke primjene obrađenog materijala u praksi.

Na osnovu svega onoga što će biti predstavljeno u ovom poglavlju, može se zaključiti da je programiranje u *VEE Pro* programskom jeziku vrlo jednostavno. Takođe, ovaj programski jezik može poslužiti kao dobra osnova za učenje onih koji se sreću prvi put sa programiranjem. Oni koji detalnije savladaju ovaj programski jezik, moći će pomoću njega da rješavaju složene tehničke probleme iz raznih oblasti. Međutim, najveća prednost ovog programa je lakoća komuniciranja sa programabilnim mjernim instrimentima i razne mogućnosti prikaza izmjerenih vrijednosti.

Pomoću ovog programa mogu se raditi i razne simulacije procesa mjerenja. Ovo može da se iskoristi za pravljenje laboratorijskih vježbi, koje bi se izvodile na računaru prije praktičnih vježbi. Na ovaj način studenti mogu bolje da se pripreme za laboratorijske vježbe. To treba da dovede do smanjenja kvarova na klasičnim mjernim instrumentima u laboratoriji.

4.1 POČETNA PALETA (*FILE*)

Osnovni elementi menija *File* dati su na slici 4.1. Meni *File* vrlo je sličan sa ovim menijem kod drugih programa iz Windows okruženja. Tako da je za početak rada sa ovim programom dovoljno poznavanje osnova rada sa nekim drugim programskim jezikom.

Opcija ***New*** koristi se za otvaranje nove prazne radne površine za pisanje programa. Pri tome je poželjno da se predhodno spasi (ako se želi) predhodno pisani program, kako te izmjene ne bi bile izgubljene.

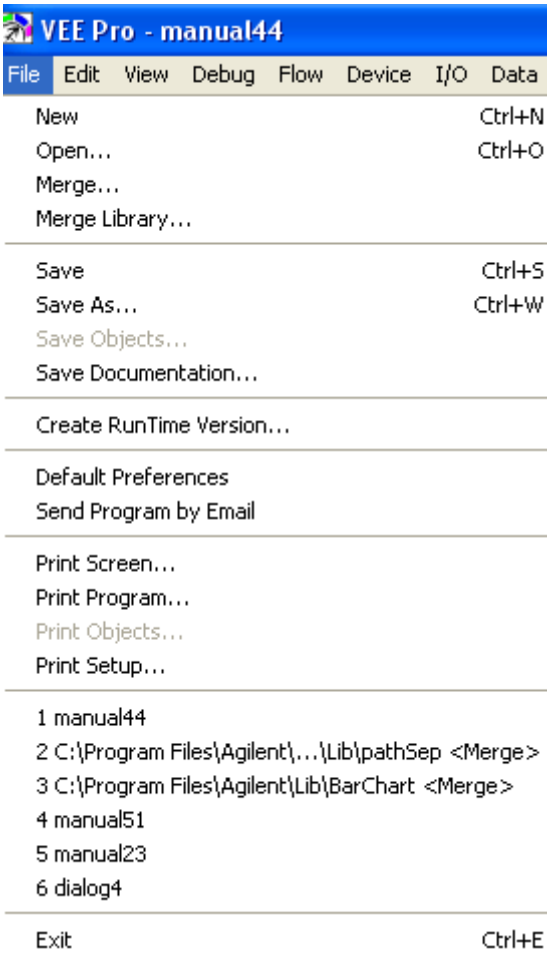
Opcija ***Open*** koristi se za otvaranje već postojećeg programa.

Opcija ***Merge*** koristi se za spajanje programa, koji se trenutno pravi sa već postojećim programom, koji je predhodno negdje spašen. Treba biti obazriv prilikom korišćenja ove opcije, jer stari program može da ima puno elemenata, koji mogu u potpunosti da prekriju naš trenutni program.

Opcija ***Save*** koristi se za spašavanje izmjena u već postojećem programu. Ovo je poželjno da se uradi nakon svake namjerno urađene izmjene u programu.

Opcija ***Save As*** koristi se za spašavanje programa pod novim imenom. Ovo je poželjno da se uradi nakon svake urađene izmjene u programu, a da pri tome ne želimo da ta promjena naruši stari program, koji je uspješno radio.

Opcija ***Save Documentacion*** koristi se za pravljenje tekstualnog dokumenta (formata .TXT) koga generiše *VEE Pro* program automatski. U ovom dokumentu će biti navedene sve bitne karakteristike (osobine - propertis) za svaki upotrijebljeni objekat trenutno aktivnog programa. Ovaj dokumet bi trebalo obavezno napraviti uvijek, kada se program konačno završi, jer može da posluži kao dobra osnova za naknadnu analizu programa i olakša eventulne izmjene u programu.



Slika 4.1 Meni File

Opcija ***Create Run Time Version*** koristi se za pravljenje izvršne (.VXE, a ne kao kod drugih programa .EXE ekstenzije) verzije programa. Ova verzija se može napraviti, samo ako je na računaru pored *Developer* instalirana i *Run Time* verzija programa. Kao i kod drugih programskih jezika poželjno je da programer kupcu isporučuje samo izvršna *Run Time* verzija, kako on ne bi mogao da vidi razvojni kod programa. Za kod programa programer troši puno vremena i svoju pamet, pa nije dobro da to bez naknade daje drugim korisnicima.

Opcija ***Default Preferences*** koristi se da za trenutno aktivan objekat, da se sve njegove osobine podese na početne (inicijalne). Na one vrijednosti koje je objekat imao prije prvog podešavanja.

Opcija ***Print Scrin*** koristi se da za štampanje na štampaču ili spašavanje kao slike (u .MDI formatu slike) trenutno vidljivog ekrana računara, sa trenutno vidljivim kodom programa.

Opcija ***Print Program*** koristi se da za štampanje na štampaču ili spašavanje kao slike (u .MDI formatu slike) trenutno aktivnog programa. Opciono se bira štampanje veza svih objekata, ali i svih osobina za objeke koji čine program.

Opcija ***Print Setup*** koristi se da za štampanje na štampaču ili spašavanje kao slike (u .MDI formatu slike) samo osobina trenutno aktivnog programa.

Opcija ***Exit*** koristi se da za konačno zatvaranje trenutno aktivnog *VEE Pro* programa.

Na kraju menija *File* stoji lista zadnje korišćenih *VEE Pro* programa, iz koje se takođe može izvršiti pokretanje željenog programa.

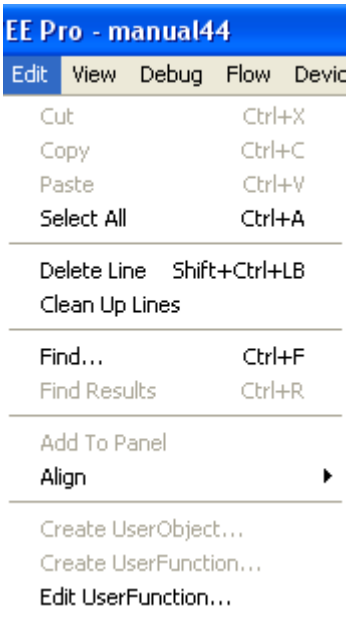
4.2 UREĐIVANJE PROGRAMA (*EDIT*)

Osnovni elementi menija *Edit* dati su na slici 4.2. Meni *Edit* jednim dijelom je vrlo sličan sa ovim menijem kod drugih programa iz Windows okruženja. Komande iz ovog menija imaju i odgovarajuće ikonice u paleti sa alatima, preko kojih se ove komande mogu brže aktivirati. Svaka od komandi u ovom editoru ima skraćenicu, preko koje se ove komande mogu pokretati i preko tastature.

Opcija ***Cut*** koristi se za isijecanje nekog markiranog objekta (ili markirane grupe objekata), koji se nalazi u programu. Pri tome isječeni objekat ostaje u pomoćnoj memoriji sve dok se ne uradi isijecanje nekog novog objekta ili se iskopira isječeni objekat sa naredbom *Paste*. Prilikom kopiranja više objekata, kopiraju se i sve linije koje predstavlaju veze između tih objekata.

Opcija ***Copy*** koristi se za kopiranje nekog markiranog objekta (ili markirane grupe objekata), koji se nalazi u programu. Pri tome isječeni objekat ostaje u pomoćnoj memoriji sve dok se ne uradi kopiranje nekog novog objekta ili se iskopira isječeni objekat sa naredbom *Paste*.

Opcija ***Paste*** koristi se za kopiranje u program nekog objekta (ili grupe objekata), koji se predhodno smjestio u pomoćnu memoriju sa komandom *Cut* ili *Copy*. Komanda *Paste* se može uzastopno koristiti više puta i na taj način jedan objekat se može uzastopno kopirati više puta.



Slika 4.2 Meni Edit

Opcija ***Select All*** koristi se za markiranje svih objekata, koji čine jedan program, pri čemu se markiraju i sve linije koje povezuju te objekte.

Opcija ***Delete Line***  koristi se za brisanje linije koje povezuju dva objekta u programu. Ikonica koja predstavlja kraticu za ovu komandu su makazice koje presijecaju liniju. Da bi neka linija bila izbrisana, potrebno je prvo kliknuti na ovu ikonicu (nakon čega se kursor miša pretvara u makazice), a zatim pozicionirati miša na liniju koju želimo ukloniti i uraditi lijevi klik miša. Pokazivač miša se može pretvoriti u makazice, ako se istovremeno pritisnu tipke *Shift* + *Ctrl*, a lijevim klikom miša se zatim brišu linije između objekata programa.

Opcija ***Find*** koristi se kao pomoć za pronalaženje, prema imenu, objekta koji postoji u programu. Ovo je posebno korisna opcija kada se nesmotrenim pomjeranjem miša po radnoj površini izgube svi objekti sa radne površine.

Opcija ***Add To Panel*** predstavlja komandu, koja je vrlo specifična za *VEE Pro* program. Pomoću ove komande se markirani objekti (ali ne i linije koje ih povezuju) prebacuju na poseban ekran. Ovaj ekran će biti vidljiv kada program bude pokrenut sa komandom *Run*. Na ovaj ekran se postavljaju samo oni objekti, koji su neophodni za pokretanje i upravljanje programom, kao i objekti za prikaz rezultata. Na ovom ekranu (panelu) se svim objektima može dodatno mijenjati pozicija, veličina, boja, kao i ostali paramatri za podešavanje, a da to ne utiče na izvršenje programa. Objekti se mogu i brisati iz panela, a da pri tome neće biti izbrisani iz koda programa.

Opcija ***Align*** koristi se za poravnavanje više markiranih objekata.

Opcija ***Create UserObject*** koristi se pravljenje novog objekta u programu od objekta, koji je trenutno markiran u programu.

Opcija ***Create UserFunction*** koristi se pravljenje nove funkcije u programu od objekta (ili više markiranih objekata), koji je trenutno markiran u programu. Na mjestu markiranog objekta se automatski pojavi objekat *Coll Ime nove funkcije*, a u program eksploreru se automatski pojavi novonastala funkcija uz funkcije koje su ranije postojale.

Opcija ***Edit UserFunction*** koristi se izmjenu sadržaja već postojeće funkcije. Ova kao i sve predhodne komande u paleti *Edit* mogu se pokreneti tako što se uradi desni klik miša nad objektom, u kome se želi uraditi neka izmjena.

4.3 PRIKAZ ELEMENATA PROGRAMA (*VIEW*)

Osnovni elementi menija *View* dati su na slici 4.3. U ovom meniju se bira način prikaza pojedinih elemenata programa. Mogu da se prikazuju promjenljive, koje su deklarisane u programu, greške koje se pojavljuju u toku izvršenja programa. Ovdje se bira da paleta za prikaz alata bude vidljiva ili sakrivena u toku procesa pisanja programa.

Opcija ***Variables*** koristi se za pronalaženje i prikaz svih deklarisanih promjenljivih, koje postoje u programu.

Opcija ***Last Error*** koristi se za pronalaženje zadnje greške prilikom izvršenja programa.

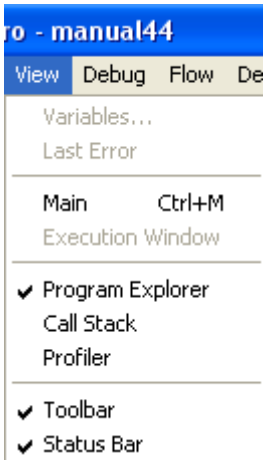
Opcija ***Main*** koristi se za zatvaranje funkcije ili objekta u kome se trenutno programer nalazi i prelazak u glavni program, koji nosi oznaku *Main*.

Opcija ***Program Explorer*** koristi se za prikaz tekućeg program eksplorera sa desne strane radne površine u kojoj se piše program.

Opcija ***Last Error*** koristi se za pronalaženje zadnje greške prilikom izvršenja programa.

Opcija ***Toolbar*** koristi se kako bi paleta sa skraćenim ikonicama (alatima) bila vidljiva ispod palete menija.

Opcija ***Status Bar*** koristi se, kako bi bila vidljiva statusna linija na dnu radnog prozura programa *VEE Pro*.



Slika 4.3 Meni View

4.4 TESTIRANJE PROGRAMA (*DEBUG*)

Osnovni elementi menija *Debug* dati su na slici 4.4. U ovom meniju se nalaze komande pomoću kojih se vrši pokretanje i zaustavljanje izvršenja pograma u procesu testiranja programa. Program se može izvršavati u cjelini, po dijelovima ili korak po korak.

Opcija ***Run*** koristi se za pokretanje kompletnog programa. U paleti alatki ovoj komandi odgovara ikonica ►.

Opcija ***Pause*** koristi se za privremeno zaustavljanje kompletnog programa. Zaustavljeni program se može ponovo pokrenuti sa komandom *Run*. U paleti alatki ovoj komandi odgovara ikonica ▌▌.

Opcija ***Stop*** koristi se za trajno zaustavljanje kompletnog programa. U paleti alatki ovoj komandi odgovara ikonica ■.

Opcija ***StepInto, Step Over*** i ***StepOut***  su tri slične komande koje se koristie za izvršenje svake komande u programu pojedinačno, odnosno objekat po objekat. Koriste se prilikom kompajliranja programa, kako bi se uočile eventualne greške u programu. Kada se pokrene ova komanda objekat koji se sljedeći izvršava biće uokviren žutom bojom. U paleti alatki ovim komandama odgovaraju ikonice.

Opcija ***Toggle Breakpoint*** koristi se za postavljanje tačke prekida programa, na objektu koji je trenutno markiran. Kada se pokrene izvršenje programa sa *Run* onda se program izvršava samo do ove tačke prekida programa. Da bi se nastavilo dalje izvršenje programa potrebno je ponovo pokrenuti program sa *Run*.

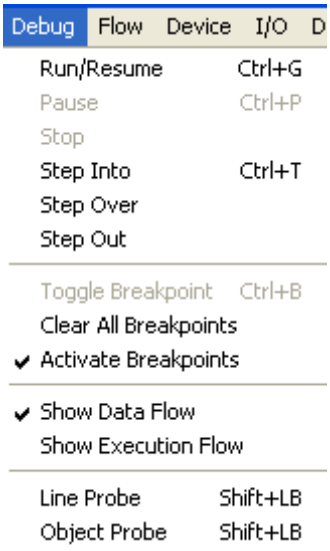
Opcija ***Clear All Breakpoints*** koristi se za uklanjanje svih postavljenih tački prekida programa u programu.

Opcija ***Activate Breakpoints*** koristi se za aktiviranje (ili uklanjanje) tački prekida programa.

Opcija ***Show Data Flow*** koristi se za prikaz toka izvršenja programa. Kockica, koja pokazuje tok izvršenja programa, se kreće između objekata programa, onako kako teče izvršenje programa.

Opcija ***Show Execution Flow*** koristi se za prikaz toka izvršenja programa. Kako teče izvršenje programa, tako svi objekti koji su se izvršili dobijaju zeleni okvir.

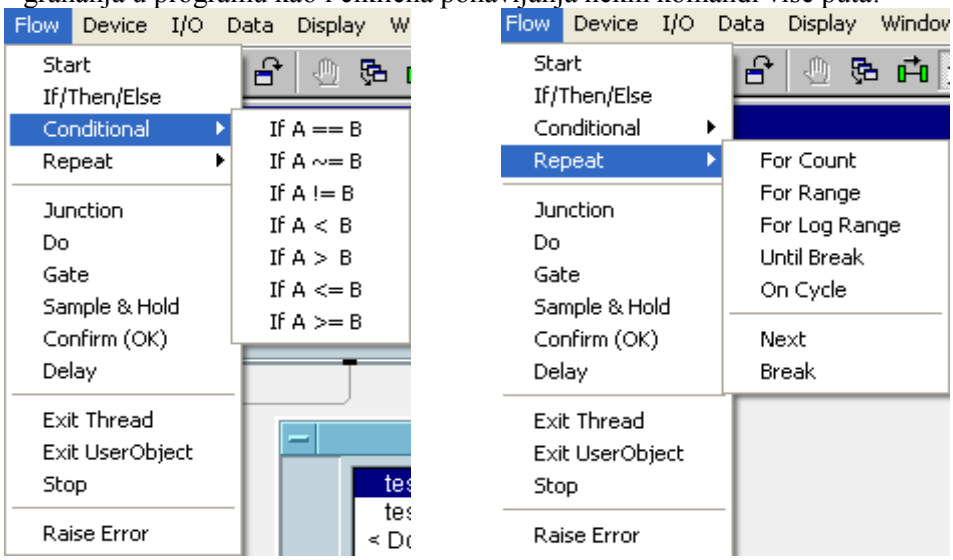
Zadnje dvije komande koje se koriste za praćenje toka izvršenja programa, mogu se koristititi u kombinaciji sa komandama ***Step Into***. Na ovaj način se bolje prati izvršenje programa korak po korak.



Slika 4.4 Meni Debug

4.5 KRETANJE KROZ PROGRAM (*FLOW*)

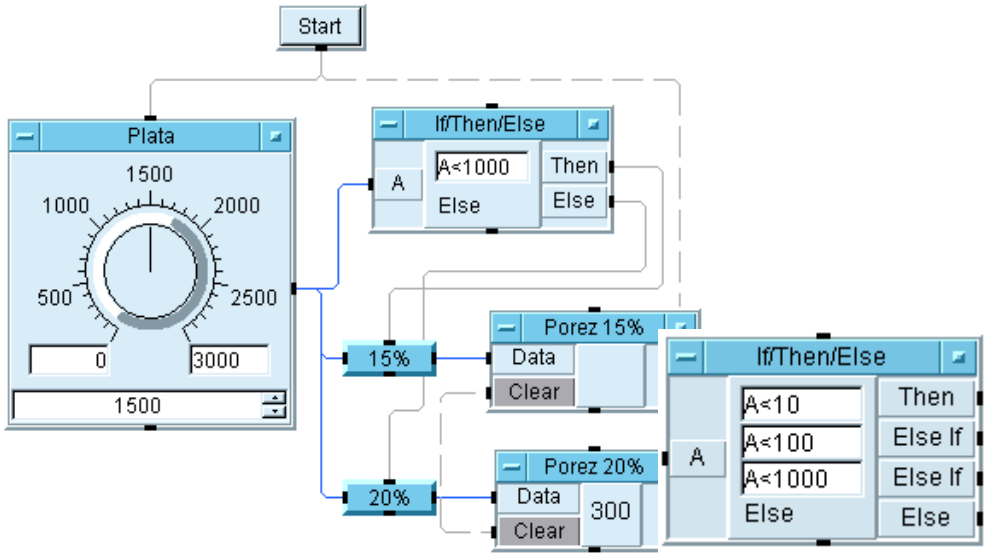
Osnovni elementi menija *Flow* dati su na slici 4.5. U ovom meniju se nalaze objekti pomoću kojih se vrši pokretanje, zaustavljanje izvršenja pograma, uslovna grananja u programu kao i ciklična ponavljanja nekih komandi više puta.



Slika 4.5 Meni Flow

4.5.1 Pokretanje programa (*Start*)

Objekat *Start* se koristi za pokretanje dijela ili kompletnog programa. Ovaj objekat ima samo jedan izlaz, koji se koristi za pokretanje drugih objekata. Klikom na ovo dugme počinje izvršenje programa. U primjeru programa prikazanom na slici 4.6, pokazano je kako se može koristiti ovaj objekat.



Slika 4.6 Primjer VEE Pro programa

Slika 4.7 If/Then/ElseIf/ElseIf

4.5.2. Grananje u programu (*If/Then/Else*)

Objekat *If/Then/Else* se koristi za grananje u programu. Ovaj objekat ima dva ulaza (jedan ulaz A se koristi za dolazak signala koji se uslovno provjerava, a drugi je kontrolni ulaz na vrhu objekta gdje se dovodi signal za start objekta) i 2 izlaza (logički izlazi koji mogu imati vrijednost 1 *True* i 0 *False*). Jedan izlaz (***Then***) daje izlaznu vrijednost 1, ako je uslov naveden u objektu ispunjen (tada je vrijednost na izlazu *Else null*). Drugi izlaz (***Else***) daje izlaznu vrijednost 0, ako uslov naveden u objektu nije ispunjen (tada je vrijednost na izlazu *Then null*). U primjeru programa prikazanom na slici 4.6, pokazano je kako se može koristiti ovaj objekat.

Pomoću ovog objekta se može napraviti i struktura *If/Then/ElseIf/ElseIf/Else* kao na slici 4.7. Ova struktura omogućuje da se nad jednim ulazom A postavlja više različitih uslova, koji su međusobno u suprotnosti. Pri upotrebi ovog objekta treba voditi računa da uslovi budu postavljeni po rastućem redoslijedu, kako se ne

bi međusobno isključivali. Da bi se dobila ova struktura od standardnog objekta *If/Then/Else*, potrebno je prvo dodati ovaj objekat na radnu površinu za pisanje programa. Zatim uraditi desni klik miša nad ovim objektom i iz menija koji se pojavi izabrati opciju *Insert Else/If*. Ovaj postupak ponavljati onoliko puta koliko se želi postaviti različitih uslova.

4.5.3 Grananje poslije poređenja (*Conditional*)

Objekat *Conditional* se koristi za grananje u programu nakon poređenja dva ulazna signala. Postoji 7 različitih varijanti ovog objekta. Svaka varijanta ima obavezno 2 ulaza za signale koji se porede (A i B). Uslov poređenja dva signala može da se piše na sljedeće načine:

- A == B uslov ispunjen kada su vrijednosti A i B iste.
- A > B uslov ispunjen kada je vrijednost A veća od vrijednosti B.
- A < B uslov ispunjen kada je vrijednost A manja od vrijednosti B.
- A >= B uslov ispunjen kada je vrijednost A veća ili jednaka B.
- A <= B uslov ispunjen kada je vrijednost A manja ili jednaka B.
- A != B uslov ispunjen kada je vrijednost A različit od vrijednosti B.
- A ~= B uslov ispunjen kada je vrijednost A približno jednaka B.

Jedan izlaz (*Then*) daje izlaznu vrijednost 1, ako je uslov naveden u objektu ispunjen. Drugi izlaz (*Else*) daje izlaznu vrijednost 0, ako uslov naveden u objektu nije ispunjen.

Uslov u ovom objektu može biti napisan i na sljedeća dva načina:

- A > 1 AND B < 10 složeni uslov u objektu je ispunjen samo, kada su ispunjena obadva uslova istovremeno i prvi (A > 1) i drugi uslov (B < 10).
- A > 1 OR B < 10 složeni uslov u objektu je ispunjen samo, kada je ispunjen jedan od dva uslova prvi (A > 1) ili drugi uslov (B < 10).

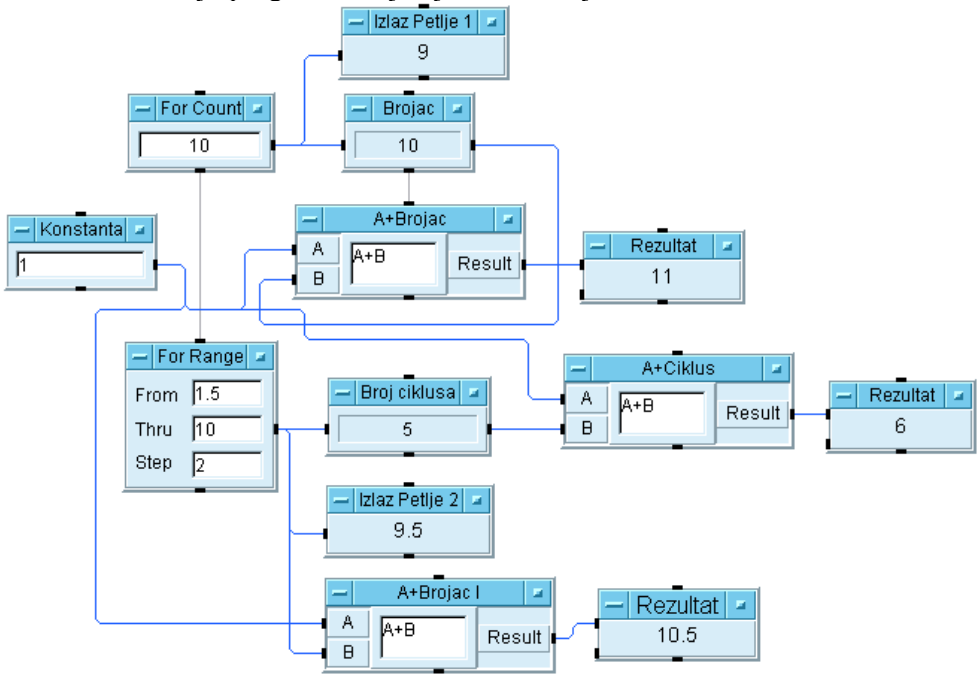
4.5.4 Višestruko ponavljanje (*Repeat*)

Objekat *Repeat* se koristi za višestruko ponavljanje nekih objekata u programu. Postoji 5 različitih varijanti ovog objekta:

- *For Count*
- *For Range*
- *For Long Range*
- *Until Break*
- *On Cycle*.

Objekat ***For Count*** namijenjen je za ponavljanje nekog bloka naredbi u programu, tačno određen broj puta sa korakom 1. Broj ponavljanja se upisuje direktno u ovaj objekat. Objekat je napravljen tako da brojanje počinje od nule (0,

1, 2, 3, ..., n-1), a završava se sa brojem za jedan manjim od onog koji je unesen u objekat. U primjeru programa prikazanom na slici 4.8, pokazano je kako se može koristiti ovaj objekat. Brojanje traje sa korakom 1, počinje od 0, a završava se sa 9. Na ovoj slici odjekat sa nazivom *Izlaz iz petlje1* predstavlja analogni displej za prikaz trenutne vrijednosti na izlazu objekta *For Count* i vidimo da na kraju programa ovaj objekat ima vrijednost 9. Objekat sa nazivom *Brojac* predstavlja brojač koji broji ukupan broj ciklusa koliko se ovaj blok naredbi ponavljao i vidimo da na kraju programa ovaj objekat ima vrijednost 10.



Slika 4.8 Objekat For Count i For Range

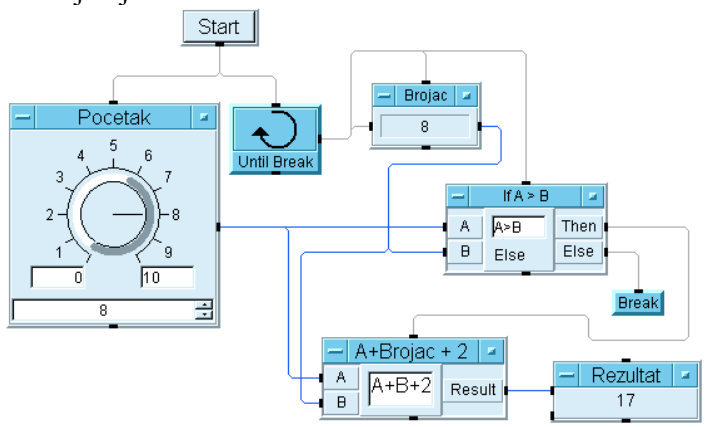
Objekat ***For Range*** namijenjen je za ponavljanje nekog bloka naredbi u programu, tačno određen broj puta sa korakom koji može biti promjenljiv. Brojanje počinje od broja koji se upisuje u polje *From*. Brojanje prestaje kada brojanje dođe do broja, koji se upisuje u polje *Thru*. Korak sa kojim se broji upisuje se u polje *Step*. Korak brojanja može biti i jedan, ali i brojevi veći i manji od jedan. U primjeru programa prikazanom na slici 4.8, pokazano je kako se može koristiti ovaj objekat. Brojanje traje 5 ciklusa sa korakom 2, počinje od 1.5, a završava se sa 9.5 (1.5, 3.5, 5.5, 7.5, 9.5). Iako je u objektu upisano da brojanje ide do 10, ovaj broj se ne dostigne nikada zbog brojanja sa korakom 2. Na ovoj slici odjekat sa nazivom *Izlaz iz petlje2* predstavlja analogni displej za prikaz trenutne vrijednosti na izlazu objekta *For Range* i vidimo da na kraju programa ovaj objekat ima vrijednost 9.5. Objekat sa nazivom *Broj ciklusa* predstavlja brojač, koji broji ukupan broj ciklusa

koliko se ovaj blok naredbi ponavljao i vidimo da na kraju programa ovaj objekat ima vrijednost 5. U ovom objektu se za početak i kraj brojanja ne mora uzeti cjelobrojna vrijednost, a mogu se uzeti i negativni brojevi (npr. From -8, Thru -1).

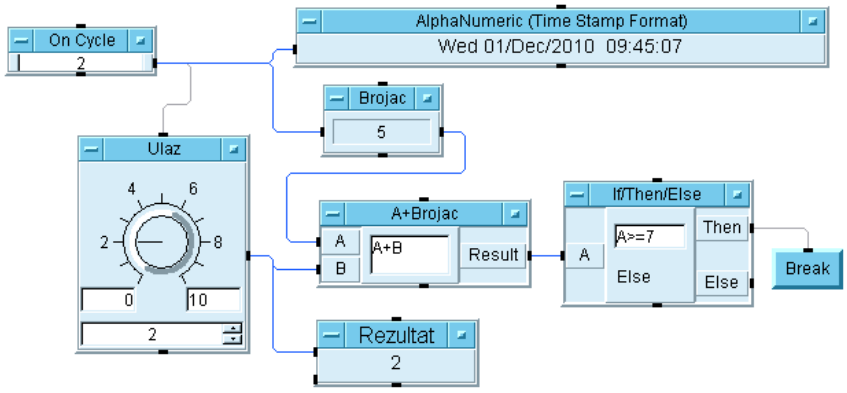
Objekat ***For Long Range*** namijenjen je za ponavljanje nekog bloka naredbi u programu tačno određen broj puta sa logoritamskim korakom koji može biti promjenljiv. Brojanje počinje od broja koji se upisuje u polje *From*. Brojanje prestaje kada brojanje dođe do broja koji se upisuje u polje *Thru*. Korak sa kojim se broji porebno je upisati u polje */Dec*. U svakom trenutku na izlazu ovog objekta postoji vrijednost koja se računa po formuli.

*Izlaz = From*exp10(n/(/Dec))*

Objekat ***Until Break*** namijenjen je za ponavljanje nekog bloka naredbi u programu sve dok se ne dogodi događaj *Break*, za prekid ponavljanja. Ako se nikada ne dogodi događaj *Break,* onda program radi u beskonačnoj petlji i neće se nikada završiti. U primjeru programa prikazanom na slici 4.9, pokazano je kako se može koristiti ovaj objekat.



Slika 4.9 Objekat Until Break

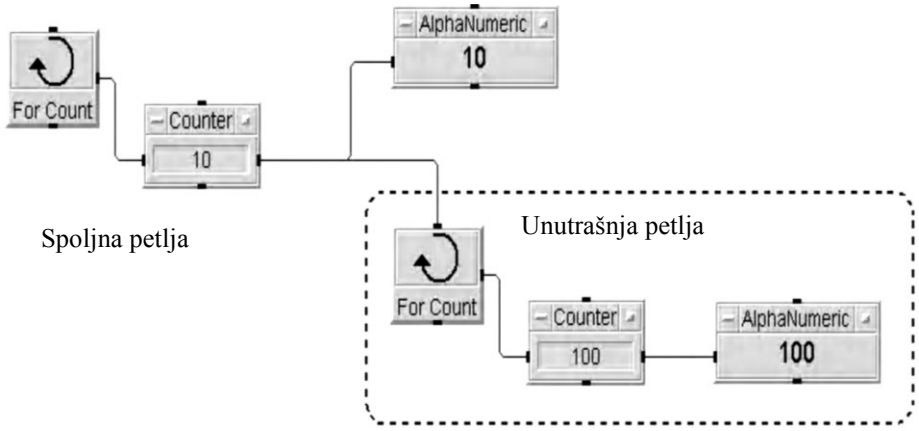


Slika 4.10 Objekat On Cycle

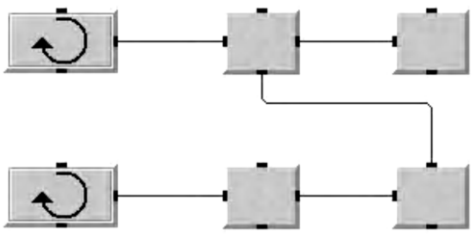
Objekat ***On Cycle*** namijenjen je za ponavljanje, u tačno definisanim vremenskim razmacima nekog bloka naredbi u programu. Ponavljanje se vrši sve dok se ne dogodi događaj *Break*, za prekid ponavljanja. Ako se nikada ne dogodi događaj *Break,* onda program radi u beskonačnoj petlji i neće se nikada završiti. U primjeru programa prikazanom na slici 4.10, pokazano je kako se može koristiti ovaj objekat. Broj koji se upisuje u ovaj objekat predstavlja vrijeme u sekundama, nakon čega se ovaj blok naredbi stalno ponavlja. U navedenom primjeru, ciklus se ponavlja svake dvije sekunde.

Prilikom korišćenja objekta *For Count* i objekta *For Range* dozvoljeno je da jedna petlja bude unutar druge kao na slici 4.11. U tom slučaju se broj ponavljanja unutrašnje petlje povećava. Ukupan broj ponavljanja unutrašnje petlje se dobije kao proizvod ponavljanja spoljašnje i unutrašnje petlje.

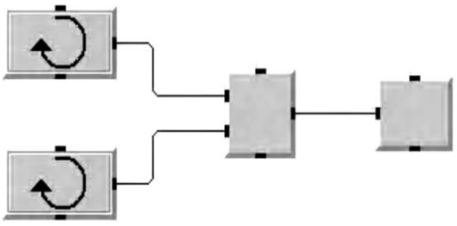
Međutim, nije dozvoljeno da se nakon završetka bloka naredbi iz jedne petlje nastavi izvršenje nekog drugog bloka naredbi, koji se nalazi unutar druge petlje, kao što je prikazano na slici 4.12. Ovo predstavlja preklapanje izvršenja dva ciklusa i program to neće dozvoliti da se izvršava. Takođe nije dozvoljeno da se sa objektom *Junction* izvrši spajanja dva objekta, koji su se izvršavali u dva različita ciklusa kao na slici 4.13.



Slika 4.11 Dozvoljeno jedna For petlja unutar druge



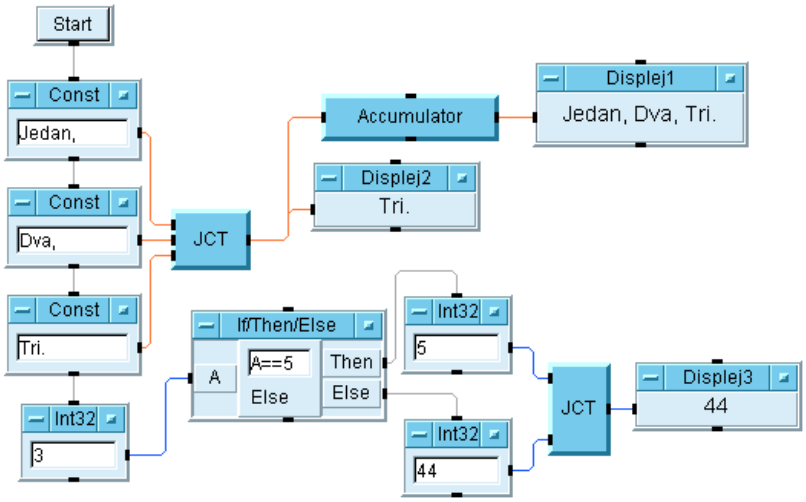
Slika 4.12 Nedozvoljeno preklapanje



Slika 4.13 Nedozvoljeno spajanje

4.5.5 Spajanje vrijednosti u programu (*Junction*)

Objekat *Junction* (JCT) se koristi za spajanje dva različita toka izvršenja programa u jedan tok. Dalje se proslijeđuje onaj tok programa koji prvi dođe do objekta *Junction*, zadnji se proslijeđuje onaj tok koji je i zadnji došao na ulaz ovog objekta. U primjeru programa prikazanom na slici 4.14, pokazano je kako se može koristiti ovaj objekat. Ovaj objekat se može koristiti za spajanje svih tipova podataka (brojeva, teksta, logičkih promjenljivih, ...).



Slika 4.14 Objekat Junction

4.5.6 Propuštanje ili zaustavljanje signala (*Do, Gate i Sample&Hold*)

Objekti *Do, Gate, Sampl&Hold, Confirm (OK) i Delay* se koriste za dalje izvršenje programa, kada na ove objekte dođe odgovarajući kontrolni signal sa vrijednošću jedan. U primjeru programa prikazanom na slici 4.15, pokazano je kako se mogu koristiti ovi objekti.

Objekat ***Do*** je najednostavniji i on propušta neki signal dalje samo, kada na njegov kontrolni ulaz (na vrhu objekta) dođe signal 1.

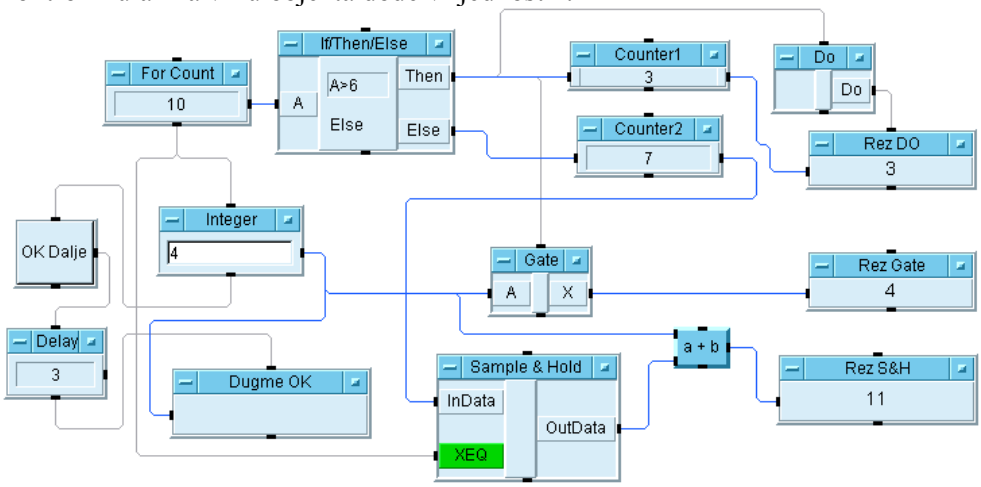
Objekat ***Gate,*** kome samo ime znači kapija. On omogućuje ili zabranjuje prolaz neke vrijednosti, koja dolazi na ulaz ovog objekta, prema izlaznom priključku ovog objekta. Prolaz je dozvoljen samo kada na kontrolni ulaz (koji se nalazi na vrhu ovog objekta) dođe signal sa vrijednošću 1. Ovaj kontrolni signal obično dolazi iz objekta *If/Then/Else*.

Objekat ***Sampl&Hold*** ima mogućnost čuvanja vrijednosti signala, koji jednom dođe na njegov ulaz. Ova vrijednost se čuva sve dok na njegov kontrolni ulaz

(XEQ) ne dođe vrijednost 1 i tada se sačuvana vrijednost proslijeđuje na izlaz ovog objekta. Ako na kontrolni ulaz nikada ne dođe signal sa vrijednošću 1, onda sačuvana vrijednost na ulazu nikada neće proći do izlaznog priključka.

Objekat ***Confirm (OK)*** je kontrolno dugme (na slici je to objekat sa nazivom OK Dalje), koje je neaktivno sve dok na njegov kontrolni ulaz na vrhu objekta ne dođe vrijednost 1. Kada dugme postane aktivno, klikom miša na njega može da se nastavi sa daljim izvršenjem programa.

Objekat ***Delay*** zaustavlja izvršenje programa na određeni vremenski period, koji je unesen u objekat (uneseni broj predstavla vrijeme u sekundama koliko će izvršenje programa biti zaustavljeno, na slici to je 3 sekunde), kada na njegov kontrolni ulaz na vrhu objekta dođe vrijednost 1.



Slika 4.15 Objekat Do, Gate, Sampl&Hold, Confirm (OK) i Delay

4.5.7 Kraj programa (*Exit Thread, Exit User Object, Stop i Raise Error*)

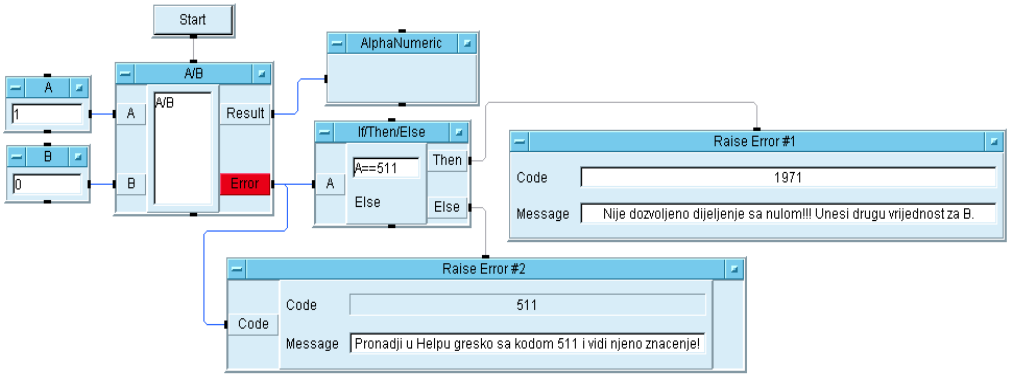
Objekti *Exit Thread, Exit User Object, Stop i Raise Error* se koriste za zaustavljanje izvršenja programa.

Objekat ***Exit Thread*** se koristi za zaustavljanje izvršenja dijela programa, kada u bilo kojoj grani programa na kontrolni ulaz ovog objekta dođe signal sa vrijednošću 1.

Objekat ***Exit User Object*** se koristi za zaustavljanje izvršenja podprograma (u *VEE Pro* se podprogrami zovu *User Object*), kada u bilo kojoj grani podprograma na kontrolni ulaz ovog objekta dođe signal sa vrijednošću 1. Prestankom izvršenja podprograma, nastavlja se izvršenje glavnog programa (*Main*), sa mjesta gdje je podprogram bio pozvan.

Objekat ***Stop*** se koristi za zaustavljanje izvršenja kompletnog programa, kada u bilo kojoj grani programa na kontrolni ulaz ovog objekta dođe signal sa vrijednošću 1. Postigne se isti efekat, kao kada se klikne na ikonicu *Stop* ■.

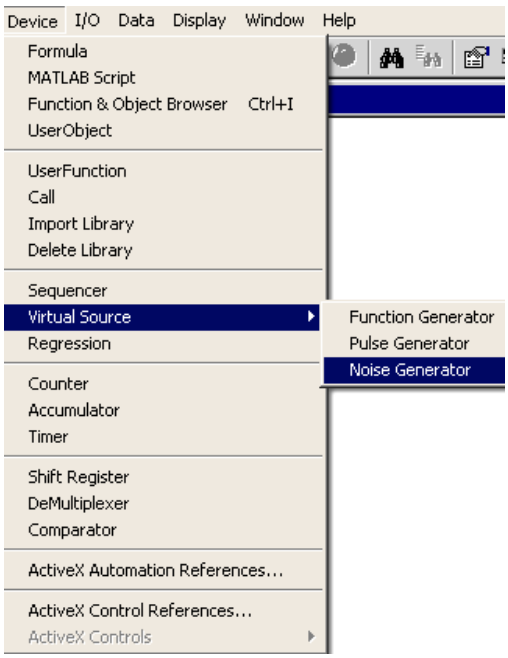
Objekat ***Raise Error*** se koristi za zaustavljanje izvršenja kompletnog programa, kada se prilikom izvršenja pojavi greška. Pomoću ovog objekta se može isprogramirati kod greške, kao i tekst poruke koji želimo da se pojavi kada greška nastane. U primjeru programa prikazanom na slici 4.16, pokazano je kako se može koristiti ovaj objekat.



Slika 4.16 Objekat Raise Error

4.6 MENI *DEVICE*

Osnovni elementi menija *Device* dati su na slici 4.17. U ovom meniju se nalaze objekti pomoću kojih se vrši pisanje formula, funkcija, procedura. U ovom meniju se nalaze i virtualni izvori pomoću kojih se mogu simulirati signali raznih oblika i frekvencija. Mogu se takođe pronaći objekti koji omogućuju brojanje ciklusa, čuvanje i spajanje podataka, kao i računanje vremena izvršenja kompletnog ili dijela programa. AktiveX kontrole za komuniklaciju sa drugim programima se takođe mogu naći u ovom meniju.

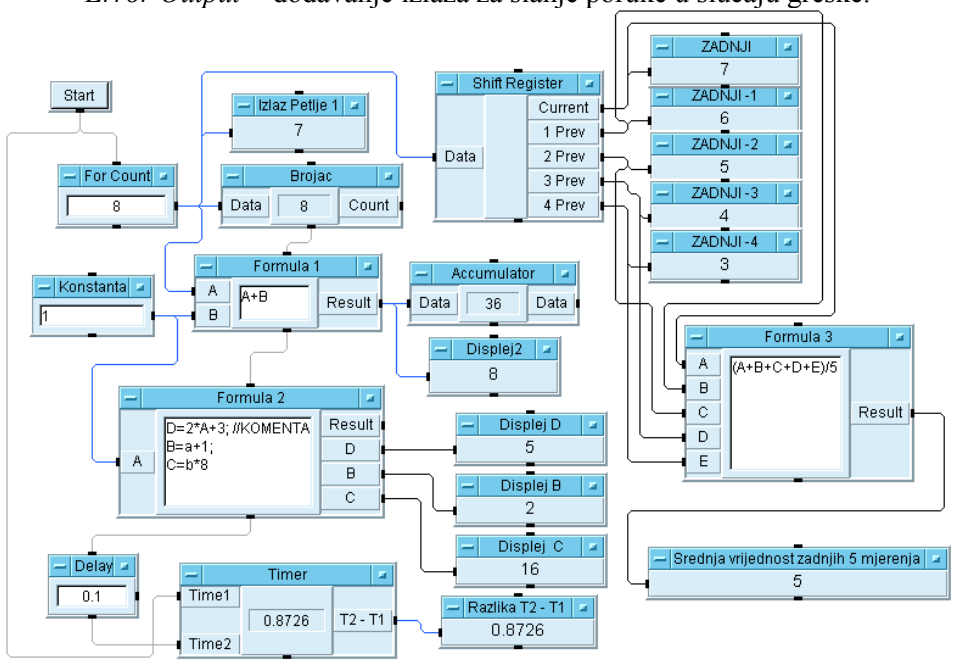


Slika 4.17 Meni Device

4.6.1 Pisanje formula u programu (*Formula*)

Objekat *Formula* je jedan od objekata, koji se najviše koristi prilikom pisanja programa u *VEE Pro*. Obično ima bar jedan ulaz i jedan izlaz, ali može imati i više ulaza kao i izlaza. U primjeru programa prikazanom na slici 4.18, pokazano je kako se može koristiti ovaj objekat. Ako objekat *Formula* ima samo jedan izlaz tada se formula piše direktno bez znaka jednako (Formula 1 na slici). Ako želimo da objekat formula ima više izlaza, tada je potrebno dodati svaki izlaz dodati posebno. Dodavanje novih izlaza se vrši tako što se prvo uradi desni klik miša nad objektom Formula. Zatim se iz padajućeg menija bira *Add Terminal* (za dodavanje novog ulaza ili izlaza) ili *Delete Terminal* (za brisanje već postojećeg ulaza ili izlaza). Kada se izabere dodavanje novog terminala onda se može izabrati jedna od opcija:

- *Data Input* - dodavanje ulaza za unos podataka.
- *Control Input* - dodavanje ulaza za kontrolne signale.
- *Data Output* - dodavanje izlaza za slanje podataka.
- *Error Output* - dodavanje izlaza za slanje poruke u slučaju greške.



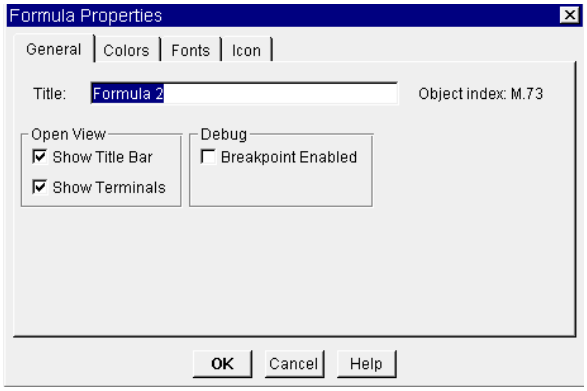
Slika 4.18 Objekat Formula, Akumulator, Brojač, Tajmer i Shift Registar

Ako objekat formula ima više izlaza, tada je potrebno prilikom pisanja formula navesti ime izlaza, znak = pa formulu koju pišemo (Formula 2 na slici). Različite formule se razdvajaju znakom ";", a komentari znakom "//".

Promjena imena objekta, ulaza, izlaza, ili podešavanje bilo koje osobine na objektu vrši se dvostrukim klikom miša na objekat, ili desni klik miša pa izbor

opcije *Propertis*. Nakon toga se pojavi prozor kao na slici 4.19, gdje se mogu podesiti opšte osobine (*General*), Boje (*Colors*), Fontovi ili naslovna ikonica. Na glavnoj paleti opcije za čekiranje imaju slijedeća značenja:

- *Title* - polje u koje se unosi tekst koji predstavlja naziv objektu.
- *Show Title Bar* - da u programu bude vidljiv naziv objekta (*Title*).
- *Show Terminals* - da u programu budu vidljivi puni nazivi ulaznih i izlaznih priključaka.
- *Breakpoint Enabled* - da se na ovaj objekat postavi kontrolna tačka prilikom pokretanja programa sa *Run*.



Slika 4.19 Prozor Propertis objekta Formula

4.6.2 Brojač (*Counter*)

Objekat *Counter* se koristi za brojanje signala, koji su dolazili na ulaz ovog objekta. U primjeru programa prikazanom na slici 4.18, pokazano je kako se može koristiti ovaj objekat.

4.6.3 Sabiranje vrijednosti (*Accumulator*)

Objekat *Accumulator* se koristi za sumiranje vrijednosti signala, koji su dolazili na ulaz ovog objekta. U primjeru programa prikazanom na slici 4.18, pokazano je kako se može koristiti ovaj objekat.

4.6.4 Računanje vremenske razlike (*Timer*)

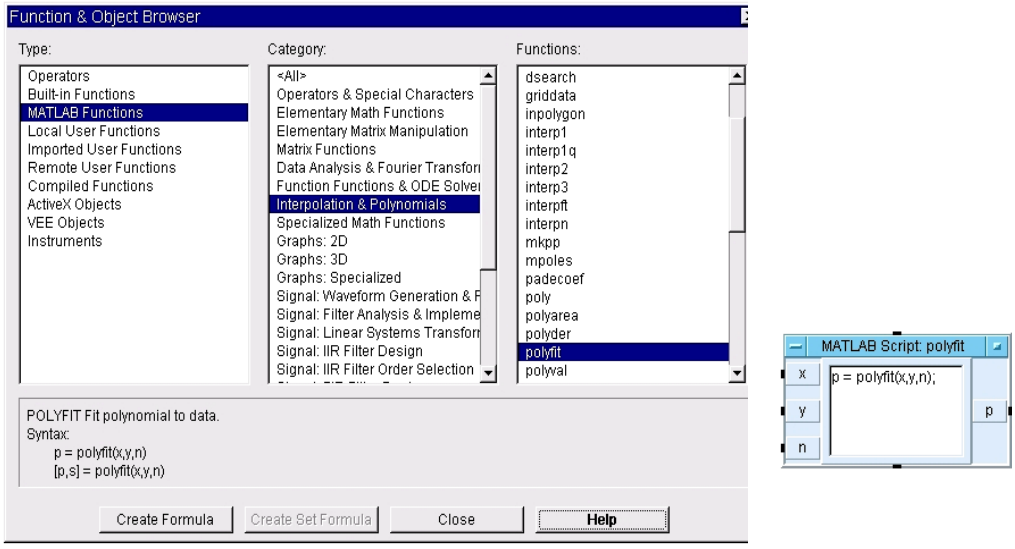
Objekat *Timer* se koristi za izračunavanje vremenske razlike između signala, koji dolaze na ulaz T1 i T2 (*vrijeme = T2 - T1*) ovog objekta. U primjeru programa prikazanom na slici 4.18, pokazano je kako se može koristiti ovaj objekat.

4.6.5 Pamćenje nekoliko zadnjih vrijednosti (*Shift Register*)

Objekat *Shift Register* se koristi za pamćenje nekoliko (dvije do deset) zadnjih *N* vrijednosti koje su došle na ulaz ovog objekta. Ovaj objekat uvijek pamti samo zadnje *N* vrijednost koje su došle na ulaz, pri čemu se uvijek najstarija *N*+1 vrijednost briše iz registra. U primjeru programa prikazanom na slici 4.18, pokazano je kako se može koristiti ovaj objekat. Predstavljen je registar koji pamti 5 zadnjih rezultata. Ovaj registar ima veliku primjenu, kada se rade praktična mjerenja sa programabilnim mjernim instrumentima i računarom. U svakom trenutku automatski se pamti nekoliko zadnjih mjerenja. Na ovaj način se u realnom vremenu smanjuje rasipanje rezultata mjerenja, koji se prikazuju na digitalnom displeju (u našem primjeru su to rezultati iz objekta *Formula 3*, koji predstavljaju u svakom trenutku srednju vrijednost od zadnjih 5 mjerenja). Što registar ima više elemenata za pamćenje, to je rasipanje rezultata manje. Na ovaj način se značajno smanjuje mjerna nesigurnost kod mjerenja.

4.6.6 *MATLAB* funkcije (*MATLAB Script*)

Objekat *MATLAB Script* predstavlja jedan od najsloženijih objekata u *VEE Pro* programu. Pruža mogućnost korišćenja velikog broja gotovih funkcija koje postoje u *MATLAB*-u. Na slici 4.20 prikazano je kako se može doći do ovih funkcija i koje su to sve funkcije. Gotove primjere programa koji koriste *MATLAB* možete naći u Helpu ovog programskog jezika, pod Examples/MATLAB.



Slika 4.20 Način izbora MATLAB funkcija

4.6.7 Gotove funkcije u *VEE Pro* (*Function&Object Browser*)

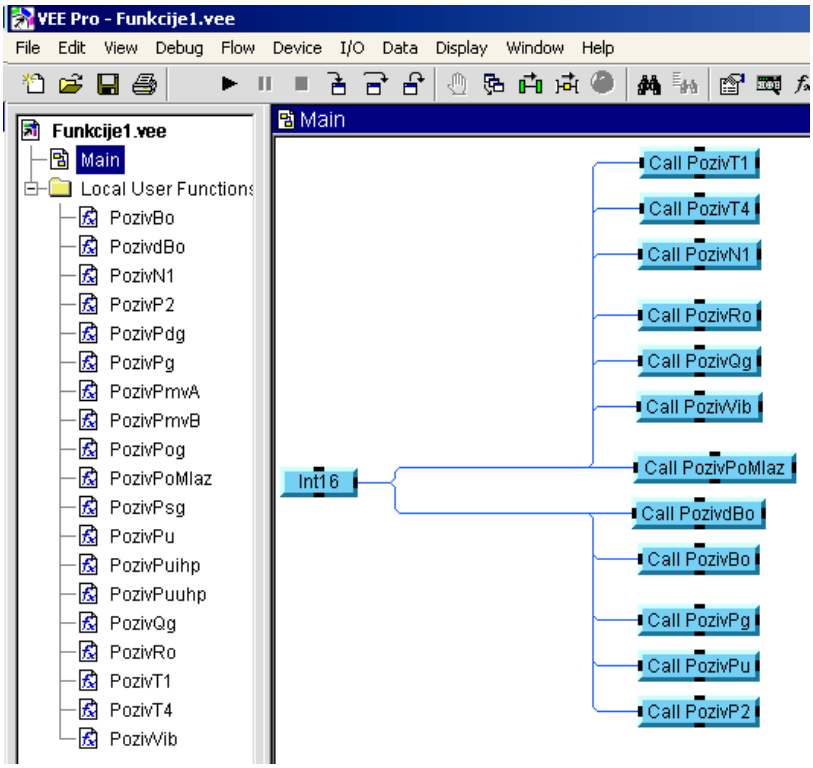
Objekat *Function&Object Browser* se bira kada se želi koristiti, neka od velikog broja gotovih funkcija u programu *VEE Pro*. Na slici 4.20 prikazano je kako se može doći do ovih funkcija i koje su to sve funkcije. Kada se pronađe željena funkcija potrebno je markirati klikom miša, a zatim kliknuti na dugme *Create Formula*. Nakon ovoga se ta funkcija dobije automatski na radnoj površini za pisanje programa, sa već pripremljenom formulum, ulazima i izlazima.

4.6.8 Pravljenje podprograma i funkcija (*UserObject, UserFunction* i *Call*)

Objekat ***UserObject*** se koristi za pravljenje podprograma u programu *VEE Pro*. Svaki podprogram ima svoje ime i svoju radnu površinu za pisanje programa.

Objekat ***UserFunction*** se koristi za pravljenje sopstvenih funkcija u programu *VEE Pro*. Svaka funkcija ima svoje ime i svoju radnu površinu za pisanje funkcije.

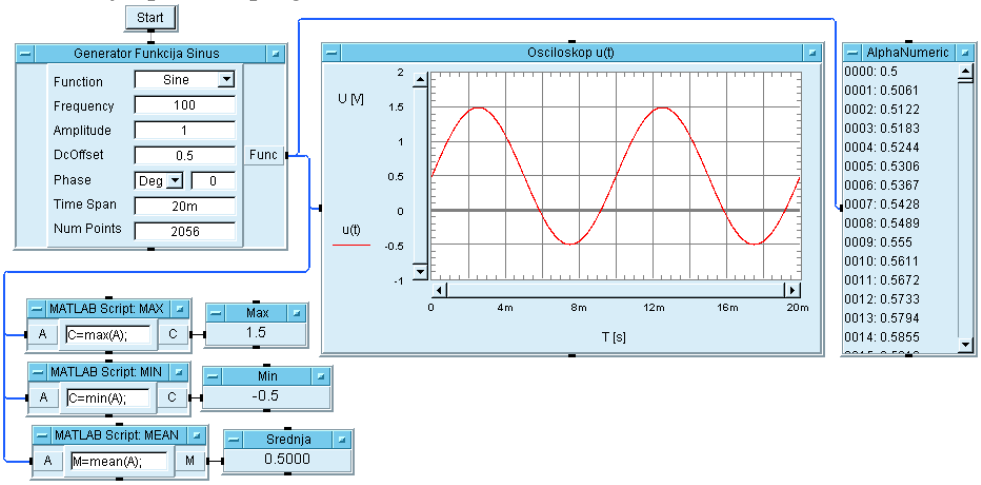
Objekat ***Call*** se koristi za poziv podprograma ili funkcije. Na slici 4.21 prikazan je program koji ima napravljeno 20 funkcija i kako se te funkcije pozivaju sa objektom *Call* iz glavnog programa *Main*.



Slika 4.21 Poziv funkcija sa Call

4.6.9 Virtuelni generatori signala (*Virtual Source*)

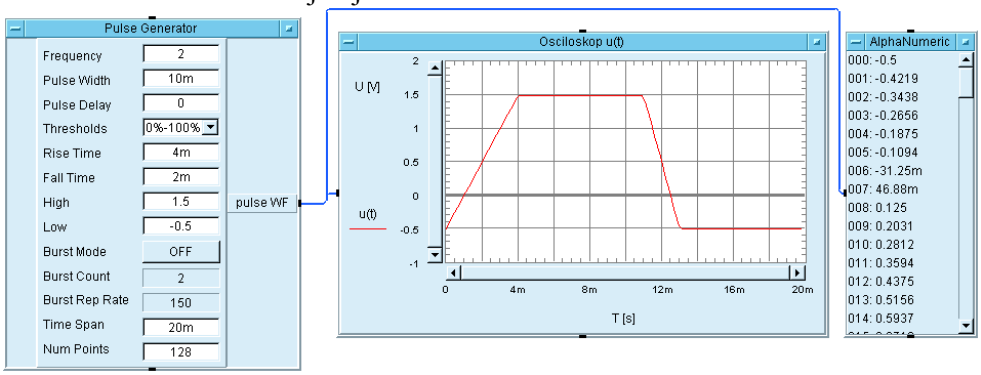
Objekat *Virtual Source* ima u sebi ugrađena tri različita virtualna generatora: *Function Generator*, *Pulse Generator* i *Noise Generator*. Ovo su vrlo korisni objekti u elektrotehnici jer omogućuju simulaciju signala raznih oblika, amplitude i frekvencije, pomoću programa *VEE Pro*.



Slika 4.22 Virtualni generator funkcija

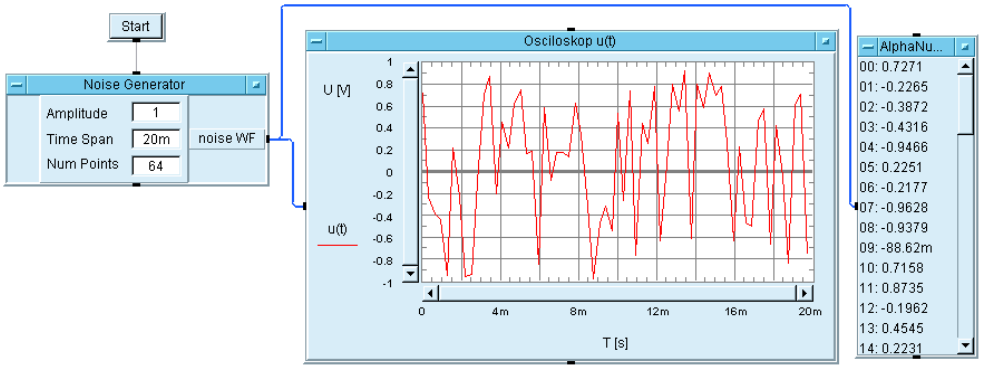
Objekat ***Function Generator*** predstavlja virtuelni generator funkcija na kome se može mijenjati oblik signala (sinusni, kosinusni, četvrtke, trouglasti i rampa) u polju *Function*, frekvencija signala u polju *Frequenci*, amplituda signala u polju *Amplitude*, početni pomjeraj signala po Y osi u polju *DC Offset*, početna fazu signala u polju *Phase,* dužinu vremenske X ose (ovo određuje broj perioda signala koje će biti generisane) u polju *Time Span* i osjetljivost generatora prema broju tačaka (32, 64, 128, ... 2056) koje čine signal u polju *Num Points*. Veći broj tačaka znači i bolji kvalitet signala, odnosno manje izobličenje. Virtualni generatori generišu signal u vidu brojnih vrijednosti zapisanih kao jednodimenzioni niz. U primjeru programa prikazanom na slici 4.22, pokazano je kako se može koristiti ovaj objekat. Prikazano je i kako se koriste *MATLAB* finkcije *Max, Min* i *Mean* za pronalaženje maksimalne, minimalne i srednje vrijednosti niza. Poželjno je da se virtualni generatori koriste u kombinaciji sa XY dijagramom (objekat *XY Trace* u *VEE Pro*) da bi mogli da posmatramo oblik generisanog signala. Kada se izlaz virtualnog generatora priključi na alfanumjerički indikator (objekat *AlphaNumjeric* u *VEE Pro*), tada se na indikatoru prikažu sve generisane vrijednosti, sa pripadajućim rednim brojem, jedna ispod druge (onoliko vrijednosti koliko je postavljeno u polju *Num Points*). Svi parametri virtualnog generatora mogu se mijenjati programski i tek tada dolaze do izražaja sve mogućnosti koje ovaj objekat pruža. Ovo se postiže dodavanjem ulaznih priključaka u virtualni generator preko opcije *Add Terminal*.

Objekat ***Pulse Generator*** predstavlja virtuelni generator impulsa, na kome se može mijenjati frekvencija signala u polju *Frequenci*, amplituda impulsa preko početka i vrha impulsa u polju *High i Low*, širina impulsa u polju *Pulse Width*, vrijeme porasta i pada impulsa u polju *Rise i Fall Time*, dužinu vremenske X ose u polju *Time Span* i osjetljivost generatora prema broju tačaka koje čine signal u polju *Num Points*. U primjeru programa prikazanom na slici 4.23, pokazano je kako se može koristiti ovaj objekat.



Slika 4.23 Virtualni generator impulsa

Objekat ***Noise Generator*** predstavlja virtuelni generator šuma na kome se može mijenjati amplituda šuma u polju *Amplitude*, dužina vremenske X ose u polju *Time Span* i osjetljivost generatora prema broju tačaka, koje čine signal u polju *Num Points*. Prilikom svakog pokretanja ovog virtualnog generatora na njegovom izlazu se generiše drugi šum po amplitudi i obliku signala. Tako da se ovaj generator ponaša i kao generator slučajnih brojeva u vidu niza brojeva. Zato se ovaj objekat može koristiti prilikom testiranja raznih programa. U primjeru programa prikazanom na slici 4.24, pokazano je, kako se može koristiti ovaj objekat u kombinaciji sa XY dijagramom i digitalnim displejom.



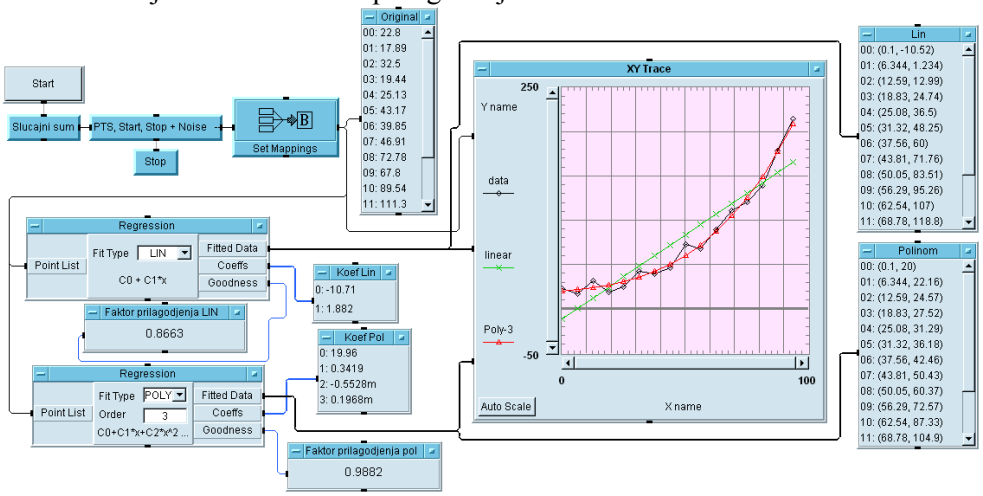
Slika 4.24 Virtualni generator šuma

4.6.10 Aproksimacija (*Regression*)

Objekat *Regression* koristi se za aproksimaciju podataka u obliku niza, koji u XY kordinatnom sistemu predstavljaju neku krivu. Pomoću ovog objekta moguće je uraditi sljedeće tipove aproksimacije:

- linearna,
- logoritamska
- eksponencijalna i
- polinomom određenog reda.

U primjeru programa prikazanom na slici 4.25, pokazano je, kako se može koristiti ovaj objekat u kombinaciji sa XY dijagramom i digitalnim displejom. Kriva označena sa *data* (crna linija sa kvadratićima) predstavlja originalnu krivu koju želimo da aproksimiramo. Kriva označena sa *linear* (zelena linija sa iksevima) predstavlja linearnu aproksimaciju. Kriva označena sa *Poly-3* (crvena linija sa trokutićima) predstavlja aproksimaciju polinomom trećeg reda. Može se uočiti da je aproksimacija polinomom trećeg reda vrlo bliska originalnoj krivoj, za razliku od linearne aproksimacije, koja ima veliko odstupanje od originalne krive. Objekat *Regression* na izlazu *Coeffs* daje izračunate koeficijente aproksimacije, a na izlazu *Goodness* daje izračunat faktor prilagođenja.



Slika 4.25 Aproksimacija krive linearno i polinomom 3 reda

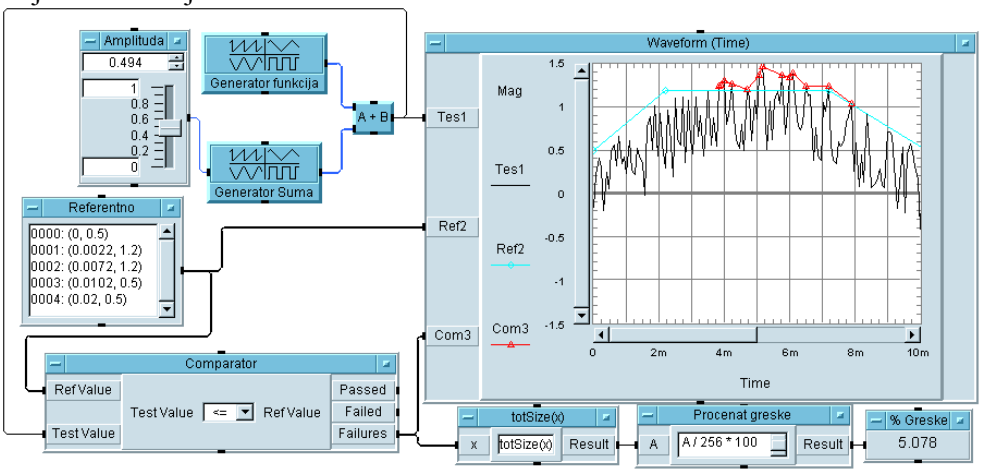
.

4.6.11 Poređenje dva signala (*Comparator*)

Objekat *Comparator* koristi se za međusobno poređenje podataka u obliku niza, koji u XY kordinatnom sistemu predstavljaju neku krivu. Pomoću ovog objekta moguće je uraditi poređenje operatorima: ==, <, <=, >, >=.

U primjeru programa prikazanom na slici 4.26, pokazano je kako se može koristiti ovaj objekat u kombinaciji sa XY dijagramom. U primjeru sa slike vršeno je

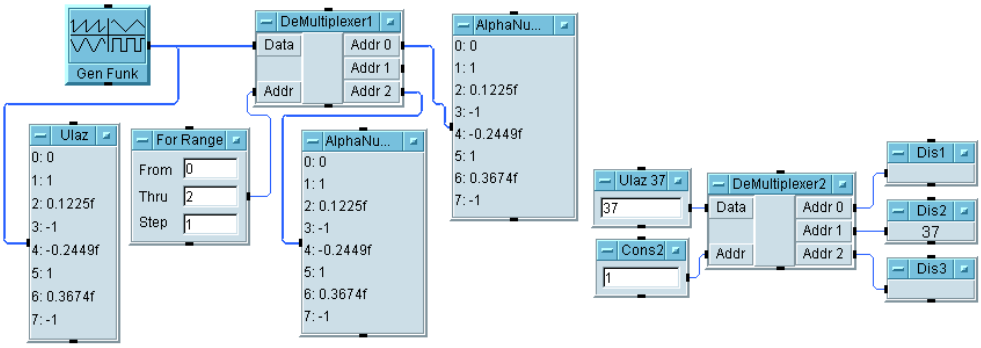
poređenje referentnog signala (koji je na dijagramu predstavljen plavom linijom sa kvadratićem i ima 5 tačaka) i signala koji predstavlja zbir signala iz generatora funkcija i generatora šuma (koji je na dijagramu predstavljen crnom linijom i ima 256 tačaka). Rezultat poređenja je kriva, koja se pojavljuje samo ako postavljeni uslov nije ispunjen (koja je na dijagramu predstavljen crvenom linijom sa trokutićima i ima 13 tačaka u kojima se desila greška). Rezultat poređenja su vrijednosti dobijene na izlazu *Failures*.



Slika 4.26 Poređenje dva signala

4.6.12 Slanje signala na više strana (*DeMultiplexer*)

Objekat *DeMultiplexer* se koristi da jedan signal istovremeno proslijedimo na dva ili više izlaza.



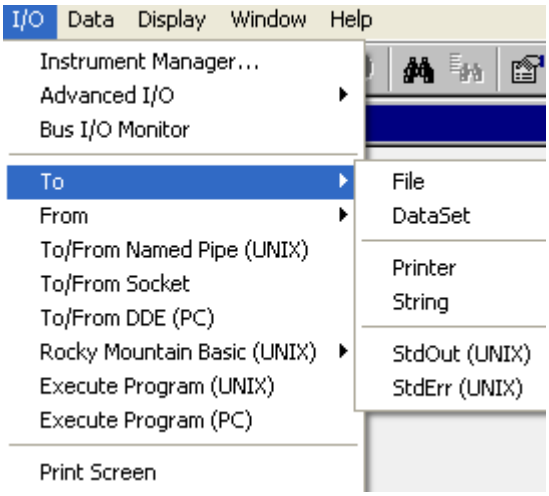
Slika 4.27 Objekat DeMultiplekder

U primjeru programa prikazanom na slici 4.27, pokazano je kako se može koristiti ovaj objekat. Na ulaz objekta *Data* se dovodi signal koji se prosljeđuje, a na ulaz *Addr* se dovodi redni broj izlaza na koji se prosljeđuje ulazni signal. U

primjeru sa slike *DeMultiplekser1* istovremeno na sva tri izlaza proslijeđuje ulazni signal, jer se istovremeno aktiviraju sve tri izlazne adrese preko *For* petlje. U primjeru sa slike *DeMultiplekser2* proslijeđuje ulazni signal samo na drugi izlaz (jer je adresa prvog izlaza 0, a drugog 1).

4.7 POVEZIVANJE UREĐAJA SA RAČUNAROM (*I/O*)

Osnovni elementi menija *I/O* dati su na slici 4.28. U ovom se meniju nalaze objekti pomoću kojih se vrši pronalaženje interfejsa preko kojih su mjerni instrumenti povezani sa računarom. U ovom meniju se nalaze i objekti pomoću kojih se mogu slati razni podaci u neki tekstualni dokument, a poslije te snimljene podatke preuzimati iz tih dokumenata. Pomoću objekta *Print Scrin* može se vršiti štampanje programa i rezultata mjerenja na štampaču ili u neki fajl kao slika.



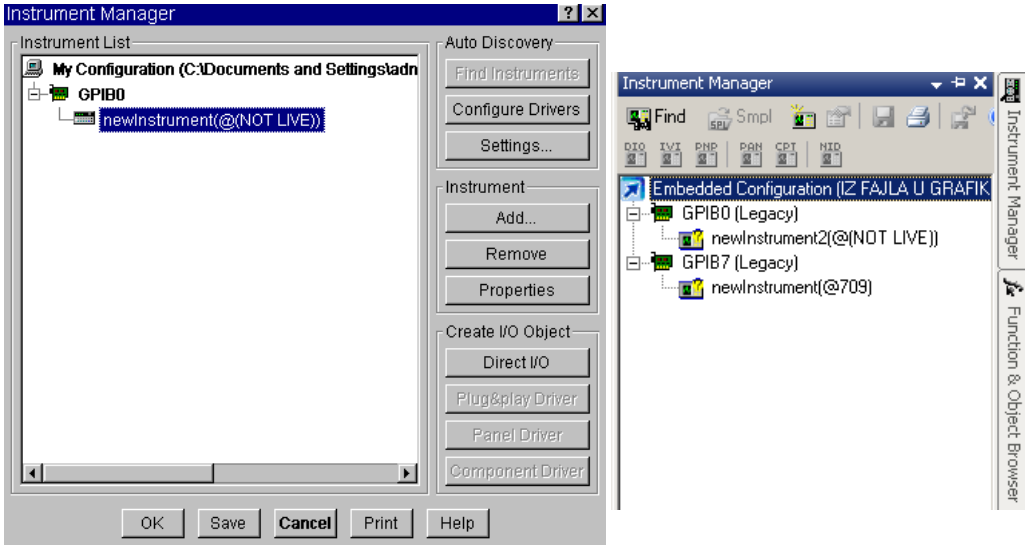
Slika 4.28 Meni I/O

4.7.1 Pronalaženje interfejsa i uređaja (*Instrument Menager*)

Objekat *Instrument Menager* koristi se prvo za pronalaženje interfejsa preko kojih se programabilni instrumenti mogu povezani sa računarom, a zatim i svih programabilnih instrumenata koji su u tom trenutku uključeni i preko interfejsa povezani sa računarom. Na slici 4.29 prikazam je prozor *Instrument Menager*, koji se koristi za pronalaženje odgovarajućeg interfejsa u verziji programa *VEE Pro* 6.0 (lijevo) i verziji *VEE Pro* 8.5 (desno). Programabilni instrumenti i uređaji se mogu spajati sa računarom preko sljedećih interfejsa:

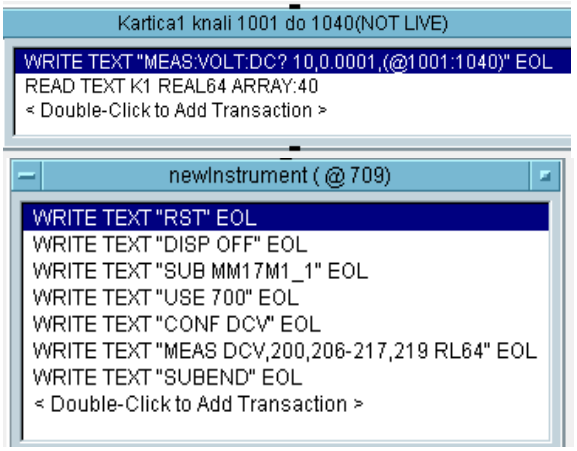
- *GPIB*
- *Serial*
- *GPIO*
- *USB/GPIB*
- *Lan* (samo za verzije programa *VEE Pro* 8.5 i novije).

Pošto se na jedan računar može istovremeno spojiti više programabilnih uređaja, onda je bitno da se dodijeli različita adresa svakom uređaju. Dodavanje objekta za komunikaciju sa tim uređajem u program vrši se za verziju 6.0 klikom na dugme *Direct I/O*, a za verziju 8.5 i novije duplim klikom na aktivni instrument.



Slika 4.29 Instrument Menager u verziji VEE Pro 6.0 i VEE Pro 8.5

Kada se na radnu površinu za pisanje programa postavi novi objekat za komunikaciju sa programabilnim uređajem u njemu nema nikakvog koda. Tada se na osnovu dokumentacije proizvođača piše posebno kod za svaku komunikaciju sa uređajem. Bilo da se radi o komandama prema uređaju (npr. izbor mjernog opsega) ili o komandama pomoću kojih se preuzimaju podaci od uređaja. Na slici 4.30 prekazan je dio koda za mjerenje napona sa dva različita uređaja.



Slika 4.30 Kod programa u instrumentu

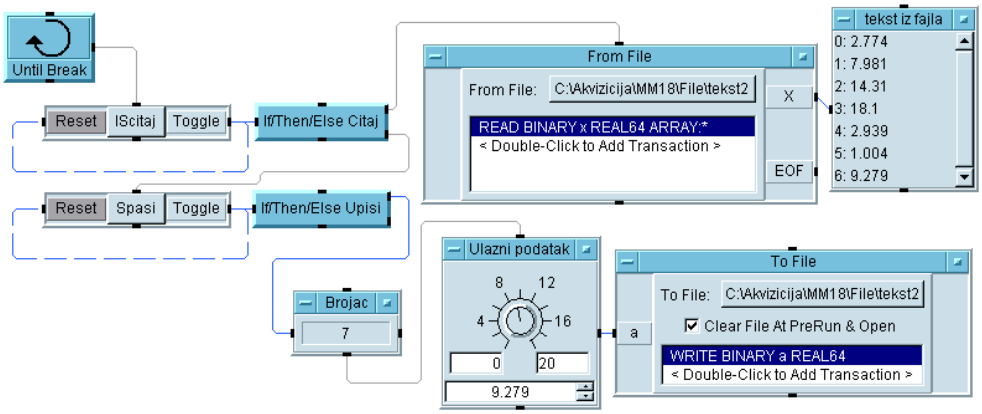
U jednom programu može postojati više objekata za komunikaciju sa istim uređajem, ali i više objekata za komuniokaciju sa različitim uređajima, koji su u istom trenutku povezani sa računarom. Pri tome treba voditi računa o redoslijedu objekata i komandama u njima, kako ne bi došlo do istovremenog slanja komandi, koje su međusobno u suprotnosti i nije ih moguće istovremeno izvršiti. Na primjer programabilni multimetar u jednom trenutku može da mjeri samo jednu veličinu, ali u malim vremenskim razmacima (koji mogu da budu reda mili sekunde) može da mjeri različite veličine (napon, struju, otpor itd.).

4.7.2 Slanje podataka (*I/O To*)

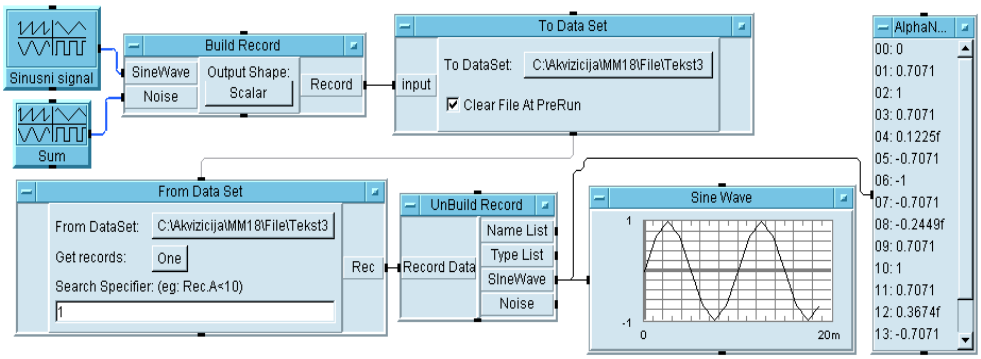
Meni ***To*** u sebi ima sljedeće objekte *File, Data Set, Printer* i *String*. Ovi objekti se mogu koristiti za slanje nekih podataka u određeni fajl, string (tekst) ili na štampač.

Objekat ***To File*** koristi se za snimanje podataka u već postojeći fajl, koji se nalazi na tačno određenoj lokaciji na računaru. Ovi fajlovi se mogu direktno čitati pomoću programa *Notepad,* a zauzimaju puno manje memorijskog prostora od *Word* ili *Excel* dokumenata.

Na slici 4.31 predstavljen je primjer programa za snimanje i čitanje podataka iz fajla. Fajl *tekst2* u koji se snimaju podaci, nalazi se na tačno definisanoj putanji "C:\Akvizicija\MM18\File\tekst2". Ako se uvijek želi snimati u prazan fajl, onda je u objektu *To File* potrebno čekirati opciju *Clear File At PreRun&Open*. Ako ova opcija nije čekirana, onda će prilikom svakog novog spašavanja podataka u fajlu ostati i sve stare snimljene vrijednosti. Pri tome se nove vrijednosti uvijek spašavaju iza starih.



Slika 4.31 Program za upis i čitanje iz fajla

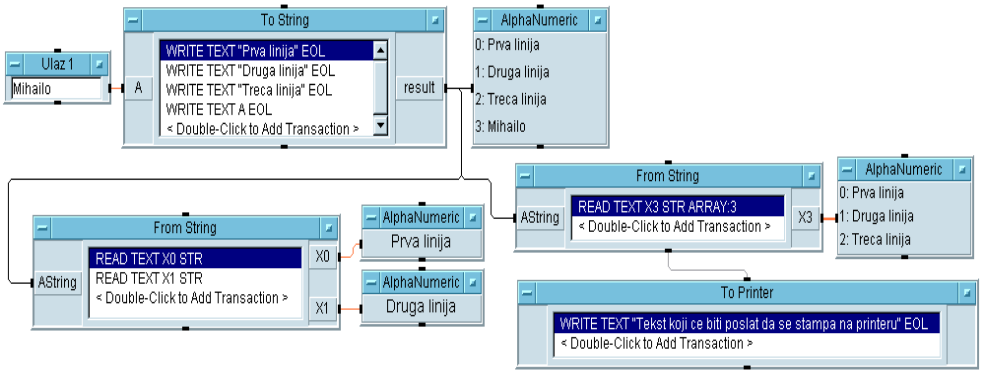


Slika 4.32 Program za upis i čitanje iz fajla pomoću Data Set

Objekat ***To Data Set*** koristi se za spašavanje podataka različitog tipa u tačno određeni fajl na poznatoj lokaciji. Da bi se u falu mogli razlikovati podaci različitog tipa i značenja, potrebno ih je pretvoriti u višedimenzioni niz (*arrey*). Na slici 4.32 predstavljen je primjer spašavanja podataka iz dva različita generatora signala u fajl *Tekst3*.

Objekat ***To Print*** koristi se za slanje tekstualnih podataka na štampač (koji je podešen da bude *difolt* na tom računaru). Na slici 4.33 predstavljen je primjer primjene ovog objekta, za štampanje teksta, koji je naveden između navodnika. *WRITE TEXT* predstavlja sistemsku komandu, koja kaže da se radi o izlaznom podatku iz ovog objekta.

Objekat ***To String*** koristi se za slanje određenih podataka u tekstualni jednodimenzioni niz (*string*). Na slici 4.33 predstavljen je primjer primjene ovog objekta, za slanje teksta, koji je naveden između navodnika, ili teksta koji u objekat dolazi preko ulaza A. Ako želimo da pošaljemo više različitih tekstualnih poruka, onda je potrebno da se više puta ponovi sistemska komanda *WRITE TEXT*.



Slika 4. 33 Program za upis i čitanje podataka u string i štampanje na štampaču

4.7.3 Prijem snimljenih podataka (*I/O From*)

Meni ***From*** u sebi ima sljedeće objekte *File, Data Set* i *String*. Ovi objekti se mogu koristiti za prijem nekih podataka iz određenog fajla ili stringa (teksta).

Objekat ***From File*** koristi se za čitanje podataka iz već postojećeg fajla, koji se nalazi na tačno određenoj lokaciji na računaru. Na slici 4.31 predstavljen je primjer jedne takve komunikacije. Fajl *tekst2,* iz koga se uzimaju podaci u ovom primjeru, se nalazi na tačno definisanoj putanji "C:\Akvizicija\MM18\File\tekst2". Iz ovog objekta podatke mogu preuzimati svi oni objekti u *VEE Pro*, koji imaju ulazne priključke za prijem podataka, isto kao da su ti podaci nastali u nekom virtualnom generatoru.

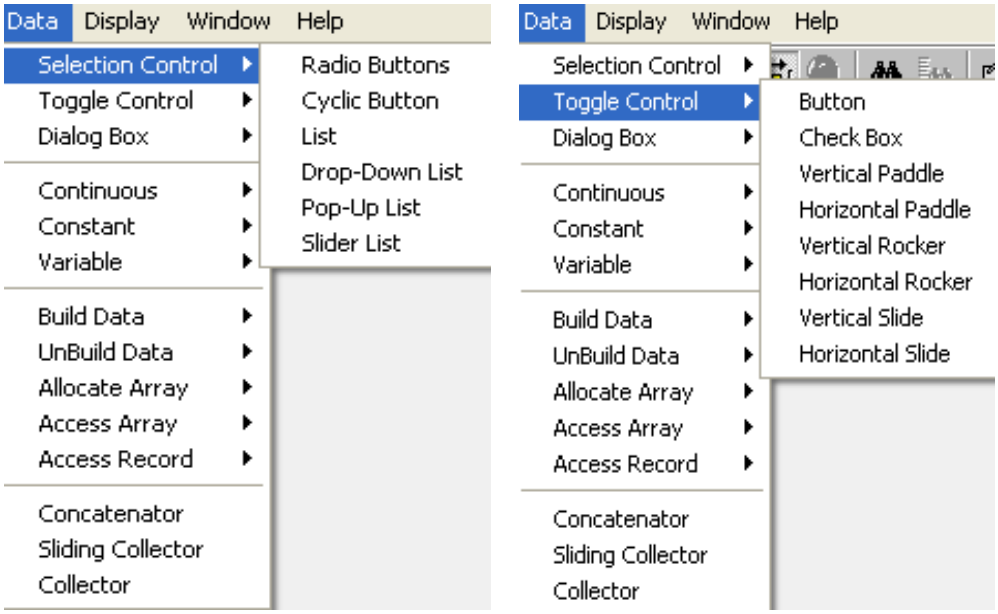
Objekat ***From Data Set*** koristi se za čitanje podataka različitog tipa iz tačno određenog fajla, koji se nalazi na poznatoj lokaciji. Da bi se iz fajla mogli

razdvojiti podaci različitog tipa i značenja potrebno ih je pretvoriti iz višedimenzionog niza (*arrey*) u jednodimenzioni niz sa samo jednim tipom podataka, a koji ujedno imaju i isto značenje. Na slici 4.32 predstavljen je primjer čitanja podataka iz jednog dokumenta *Tekst3*, koji u sebi ima podatke iz potpuno dva različita generatora signala.

Objekat ***From String*** koristi se za čitanje određenih podataka, koji su predhodno spašenu u nekom stringu. Na slici 4.33 predstavljen je primjer primjene ovog objekta, za čitanje teksta, koji je predhodno spašen u 4 reda. Čitanje podataka se radi pomoću sistemske komande "*READ TEXT ime izlaza na objektu STR" (ili "ARRAY:broj elemenata niza")*. Komanda *STR* se koristi, kada se čita samo jedan red teksta, a komanda *ARRAY* se koristi, kada da bi u jednom ciklusu pročitali više redova. Tada je potrebno striktno navesti koliko članova niza se želi pročitati.

4.8 OBJEKTI ZA UPRAVLJANJE PROGRAMOM (*DATA*)

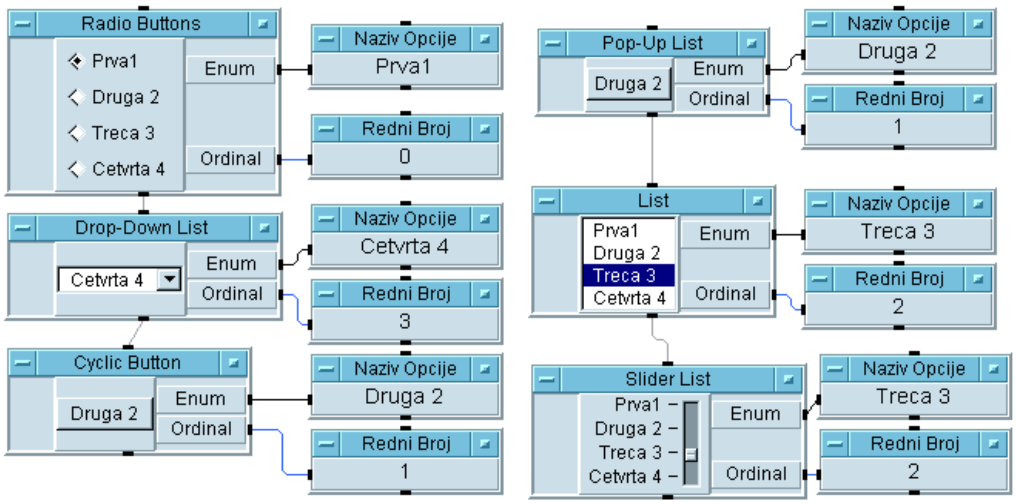
Osnovni elementi menija *Data,* dati su na slici 4.34. U ovom meniju se nalaze objekti pomoću kojih se vrši upravljanje sa programom. Kao objekti za upravljanje koriste se dugmad, izborna i opciona dugmad, liste i kombo boks. Ovdje se nalaze i objekti za kontinualno generisanje signala u vidu potenciometra ili kružnog beskonačnog dugmeta. U ovom meniju se nalaze i objekti za deklarisanje konstanti, promjenljivih i svih vrsta nizova.



Slika 4.34 Meni Data

4.8.1 Izbor jedne opcije (*Selection Control*)

Meni *Selection Control* u sebi ima šest objekata: *Radio Buttons, Cyclic Button, List, Drop-Down List, Pop-Up List i Slider List*. Ovi objekti se mogu koristiti za izbor samo jedne od više ponuđenih opcija. Primjer ovih objekata sa mogućnosti izbora jedne od 4 opcije prikazan je na slici 4.35. Svaki ovaj objekat ima 2 izlaza. Prvi izlaz *Enum* služi za slanje imena opcije. Drugi izlaz *Ordinal* šalje redni broj izabrane opcije. Redni broj prve opcije ima vrijednost nula (0), a svaka naredna je za jedan broj veća. Inicijalno ovaj objekat ima 3 opcije. Dodavanje novih opcija i promjena imena postojećim opcijama vrši se desnim klikom miša na objekat koji se mijenja. Na padajućem meniju, koji se pojavi, potrebno je izabrati opciju *Edit Enum Values*. Nakon toga se pojavi prozor kao na slici 4.36 na kome se vrši promjena imena postojećim opcijama, ili dodavanje novih opcija sa *Enter*. Potvrda izvršenih izmjena vrši se klikom na dugme *OK*.



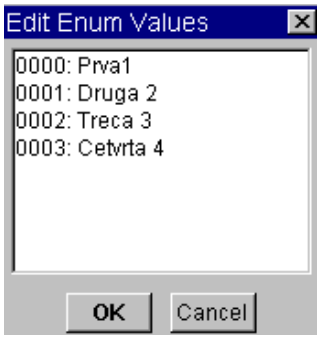
Slika 4.35 Šest objekata Selection Control

Kod objekta ***Radio Buttons*** izbor jedne opcije se vrši klikom miša na kockicu ispred imena opcije. Čim se jedna opcija izabere, predhodno izabrana opcija se automatski poništi.

Kod objekta ***Cyclic Button*** izbor jedne opcije se vrši klikom miša na dugme sa imenom opcije. Klikom na ovo dugme opcije se ciklično mijenjaju.

Kod objekta ***List*** izbor jedne opcije se vrši klikom miša na ime opcije koje se nalazi u listi.

Kod objekta ***Drop-Down List*** izbor jedne opcije se vrši klikom miša na ime opcije koje se nalazi u listi, a do liste se dolazi klikom na kombo boks .



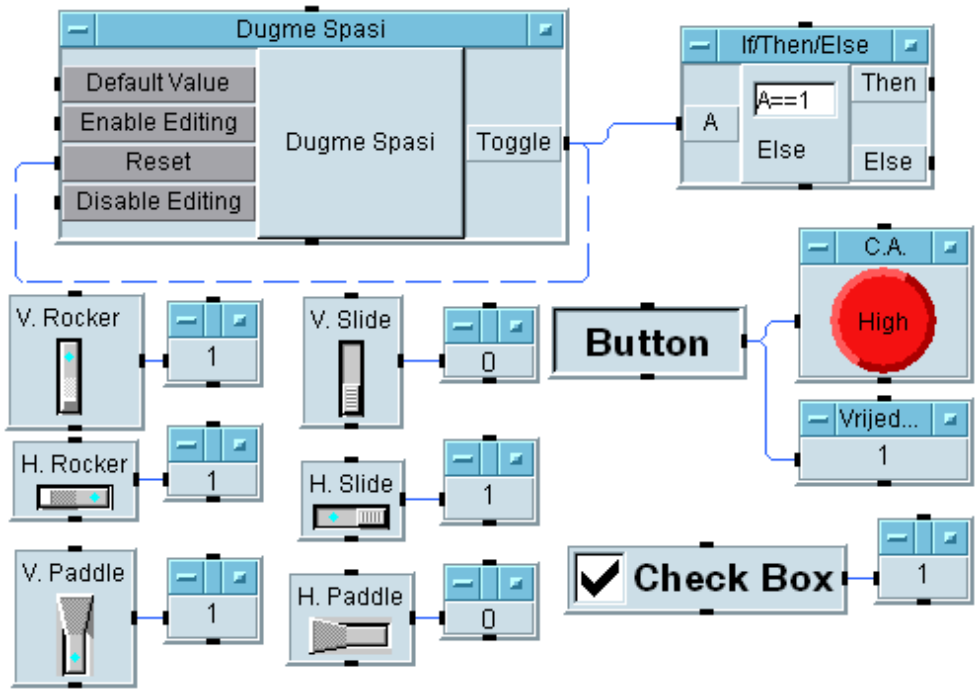
Slika 4.36 Promjena imena

Kod objekta ***Pop-Up List*** izbor jedne opcije se vrši klikom miša na dugme sa imenom zadnje, nakon čega se otvara prozor koje se nalazi lista opcija iz koje se bira jedna vrijednost.

Kod objekta ***Slider List*** izbor jedne opcije vrši se pomjeranjem klizača pored imena opcija. Kada se klizač zaustavi ostaje izabrana samo jedna opcija.

4.8.2 Komandno dugme i prekidači (*Toggle Control*)

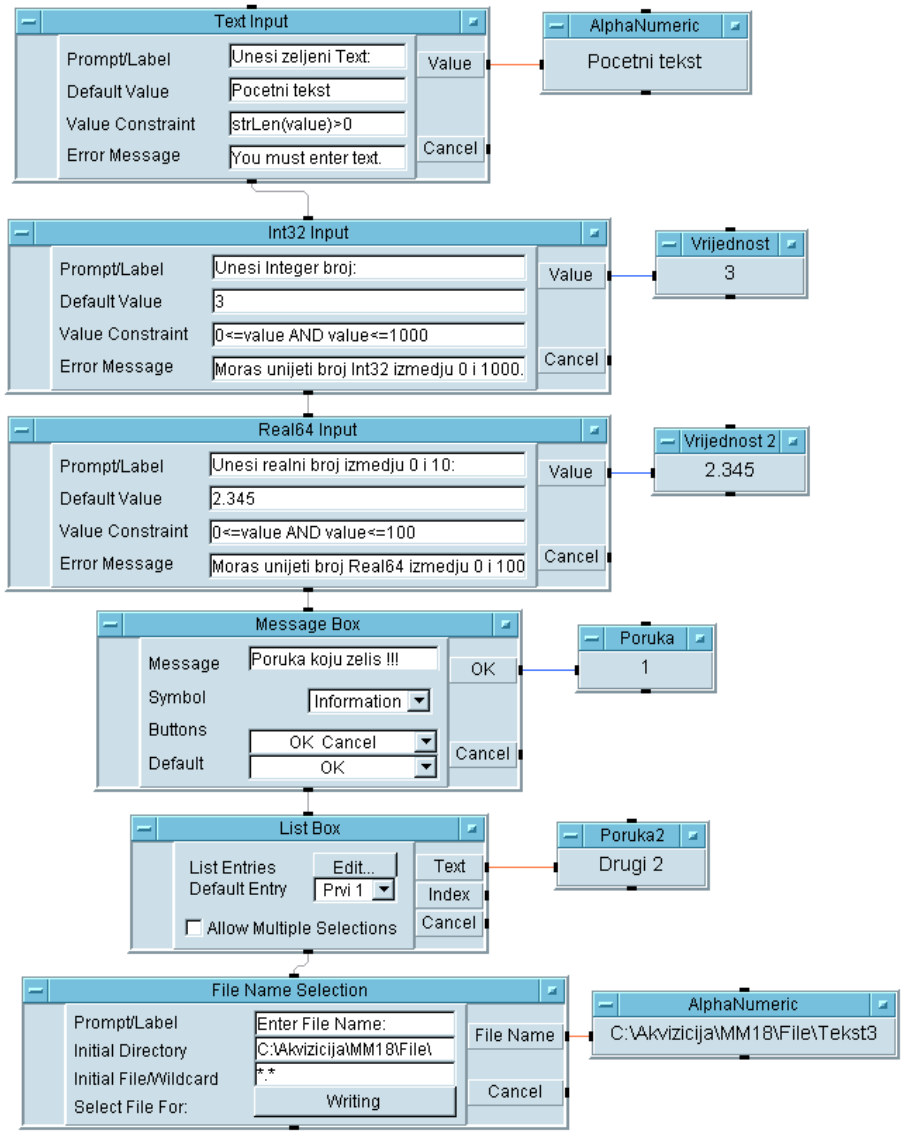
Meni *Toggle Control* u sebi ima osam objekata: *Buttons, Check Box, Vertical Paddle, Horizontal Paddle, Vertical Rocker, Horizontal Rocker, Vertical Slide i Horizontal Slide.* Ovi objekti se mogu koristiti za izbor jedne (1 - uključeno - *On*) ili druge (0 - isključeno - *Off*) opcije. Primjer ovih objekata prikazan je na slici 4.37 Ovi objekti obično idu u kombinaciji sa objektom za signalizaciju *Color Alarm* i objektom za grananje *If/Then/Else*, kako bi se utvrdilo da li je dugme pritisnuto ili ne. Objekat *Buttons* (dugme) može da ima 4 kontrolna ulaza: *Default Value* (za unos početne vrijednosti), *Enable Editing* (kada dođe vrijednost 1 omogućuje se promjena stanja dugmeta), *Reset* (kada dođe vrijednost 1 objekat se postavlja na početnu vrijednost) i *Disable Editing* (kada dođe vrijednost 1 zabranjuje se promjena stanja dugmeta).



Slika 4.37 Osam objekata Toggle Control

4.8.3 Unos brojeva i teksta (*Dialog Box*)

Meni *Dialog Box* u sebi ima šest objekata: *Text Input, Int32 Input, Real64 Input, Message Box, List Box* i *File Name Selection.* Ovi objekti se mogu koristiti za unos teksta i brojeva, javljanje poruke preko dijalog prozora, izbor neke opcije sa liste, ili izbor imena nekog već postojećeg dokumenta na računaru. Primjer ovih objekata prikazan je na slici 4.38.



Slika 4.38 Šest objekata Dialog Boxsa

Objekat ***Text Input*** koristi se za unos nekog teksta preko dijalog prozora.

Objekat ***Int32 Input*** koristi se za unos cijelih brojeva preko dijalog prozora (opseg brojeva od -2147483648 do +2147483647). U polje *Value Constraint* može se unijeti logički uslov koji broj mora da zadovolji, da bi se njegov unos prihvatio.

Objekat ***Real64 Input*** koristi se za unos realnih brojeva preko dijalog prozora.

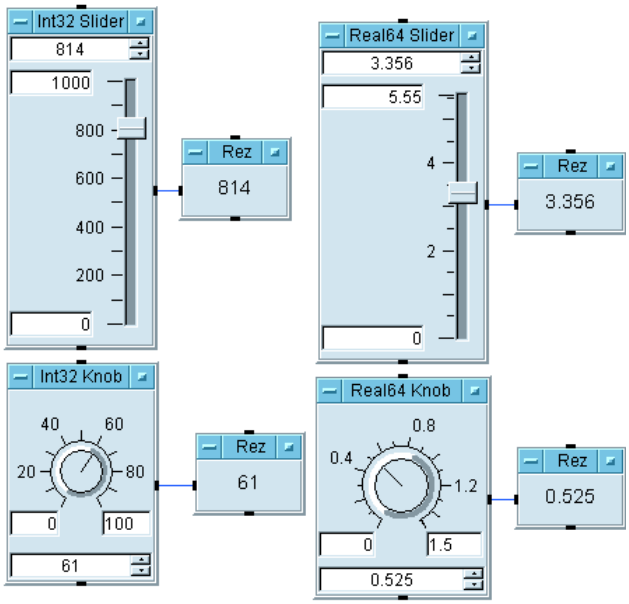
Objekat ***Message Box*** koristi se za javljanje neke poruke preko dijalog prozora. Ovaj objekat može imati šest različitih kombinacija dugmadi: **OK, OK** ili **Cancel, Abort** ili **Retry** ili **Ignore, Yes** ili **No, Yes** ili **No** ili **Cancel, Retry** ili **Cancel.**

Objekat ***List Box*** koristi se za unos nekog teksta iz ponuđene liste preko dijalog prozora. Ako je čekirana opcija *Allow Multiple Selections* onda se u listi može istovremeno čekirati više ponuđenih opcija.

Objekat ***File Name Selection*** koristi se za unos imena fajla, koji se izabere preko standardnog dijalog prozora za otvaranje dokumenata u *Windowsu*. Dijalog prozor ovog objekta može se podesiti, da se biraju oni dokumenti, koji se žele otvoriti (*Select Fille For: Reading*) ili spasiti (*Select Fille For: Writing*)

4.8.4 Potenciometar (*Continuous*)

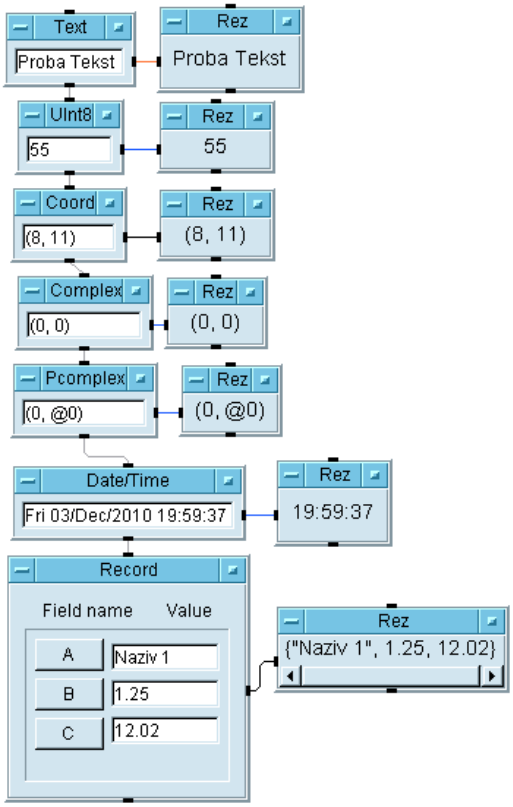
Meni *Continuous* u sebi ima četiri objekata: *Int32 Slider, Real64 Slider, Int32 Knob* i *Real64 Knob.* Oni se koristite za generisanje brojnih vrijednosti preko objekta u obliku linijskog potencimetra (*Slider*) ili objekta u vidu kružnog potencimetra (*Knob*). Promjena vrijednosti ovog objekta vrši se pomjeranjem klizača pomoću miša ili direktnim upisom brojne vrijednosti preko tastature. Primjer korišćenja ovih objekata u jednom programu prikazan je na slici 4.39. Ovi objekti mogu da imaju i više kontrolnih ulaza preko kojih se mogu podešavati svi bitni parametri ovih objekata.



Slika 4.39 Četiri objekata Continuous

4.8.5 Konstante (*Constant*)

Meni *Constant* u sebi ima trinaest objekata: *Text, UInt8, UInt16, Int16, Int32, Int64, Real32, Real64, Coord, Complex, PComplex, Date/Time* i *Record,* koji se mogu koristiti za unos konstantnih vrijednosti teksta, brojeva, kordinata kordinatnog sistema, kordinata kompleksnog broja, amplitude i faze kompleksnog broja (*Pcomplex*). Objekat za unos konstante datuma i vremena u stvari predstavlja realni broj, koji se tumači kao datum i vrijeme. Kada se ova konstanta prikazuje u alfanumjeričkom displeju potrebno je posebno formatirati ovaj objekat za prikaz datuma ili vremena. Opcija *Number* se podešava na parametar *Time Stamp* i bira se prikaz samo datuma (*Date*:), samo vremena (*Tame*:) ili i datuma i vremena zajedno (*Date&Time*:). Objekat za generisanje zapisa (*Record*) može generisati više elemenata zapisa koji mogu biti različitog tipa. Primjer ovih objekata prikazan je na slici 4.40.



Slika 4.40 Objekti constant

4.8.6 Promjenjive (*Variable*)

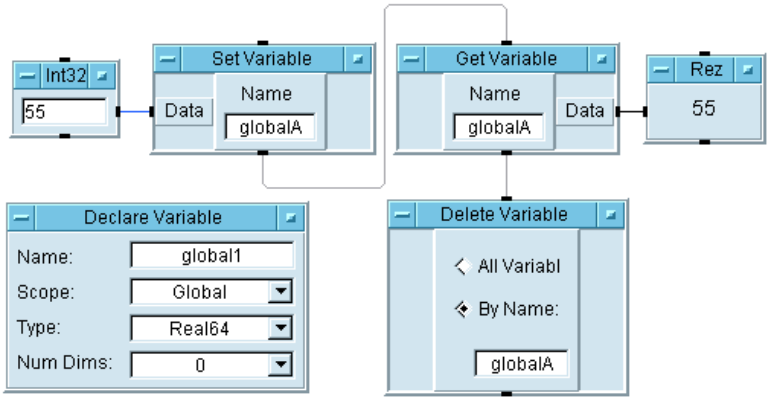
Meni *Variable* u sebi ima četiri objekata: *Set Variable, Get Variable, Delete Variable* i *Declare Variable.* Ovi objekti se koristite za rad sa promjenljivim.

Objekat ***Declare Variable*** se koristi za definisanje promjenljive u programu. Ovaj objekat stoji samostalno u programu i ne povezuje se sa drugim objektima. Svakoj promjenljivoj se obavezno daje ime u polju *Name*. U polju *Scope* se definiše da li se radi o promjenljivoj na globalnom nivou (cijeli program) ili samo na lokalnom nivou funkcije ili procedure. U polju *Type* se definiše tip podatka, koji promjenljiva može da poprimi. Kada se deklariše promjenljiva za komunikaciju sa drugim programima (*Excel, Access, Word,...*) kao tip se može navesti i objekat.

Objekat ***Set Variable*** se koristi za dodjelu vrijednosti predhodno definisanoj promjenljivoj.

Objekat ***Get Variable*** se koristi za uzimanje vrijednosti iz promjenljive, kojoj je predhodno dodjeljena vrijednost.

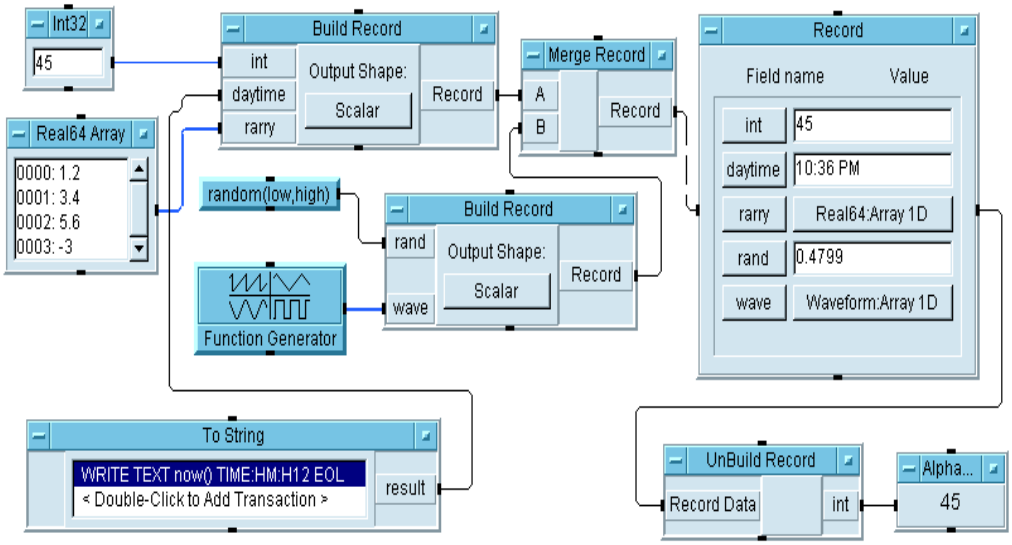
Objekat ***Delete Variable*** se koristi za brisanje sadržaja promjenljive. Primjer programa za korišćenje ovih objekata prikazan je na slici 4.41.



Slika 4.41 Objekti za rad sa promjenljiv

4.8.7 Generisanje složenih podataka (*Builde Data*)

Meni *Builde Data* u sebi ima sedam objekata: *Coord, Complex, PComplex, Waveform, Arb Waveform, Spectrum* i *Record.* Ovi objekti se koristite za generisanje složenih podataka. Primjer programa u programskom jeziku *VEE Pro* za korišćenje ovih objekata prikazan je na slici 4.42 i slici 4.43.



Slika 4.42 Objejekt za pravljenje i razdvajanje niza

Objekat ***Builde Coord*** se koristi za grupisanje podataka (po dva podatka) za XY kordinatni sistem.

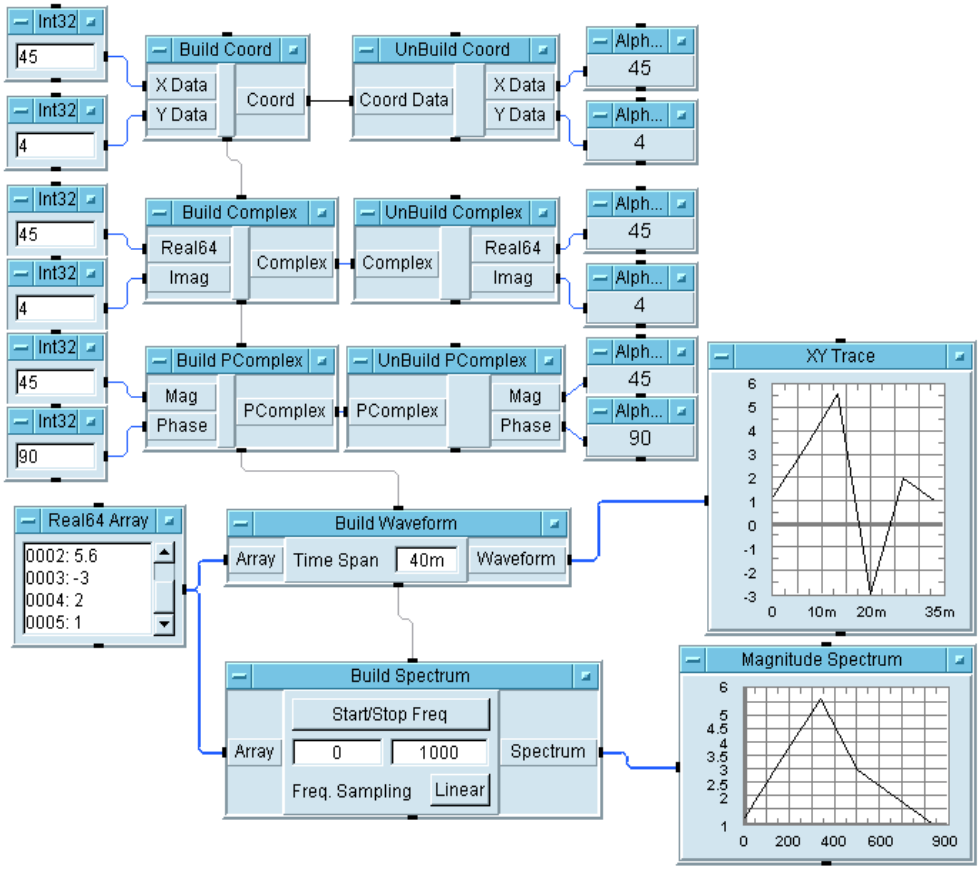
Objekat ***Builde Complex*** se koristi za pravljenje kompleksnih podataka, koji su grupisani po dva podatka. Prvi podatak predstavlja realnu, a drugi imaginarnu komponentu kompleksnog broja. Ovo je pogodno kada se radi predstavljanje naizmjeničnih signala pomoću ovog programa.

Objekat ***Builde PComplex*** se koristi za pravljenje kompleksnih podataka sa amplitudom i fazom.

Objekat ***Builde Waveform*** se koristi za pravljenje vremenski promjenljivog signala.

Objekat ***Builde Spectrum*** se koristi za pravljenje podataka, koji se prikazuju na raznim spektralnim dijagramima.

Objekat ***Builde Record*** se koristi za pravljenje zapisa (tabele -višeindeksni niz). Kretanje kroz redove tabele se ostvaruje preko dugmadi *First, Prev, Next* i *Last* (kao na slici 4.46).



Slika 4.43 Objejekti za pravljenje složenih podataka

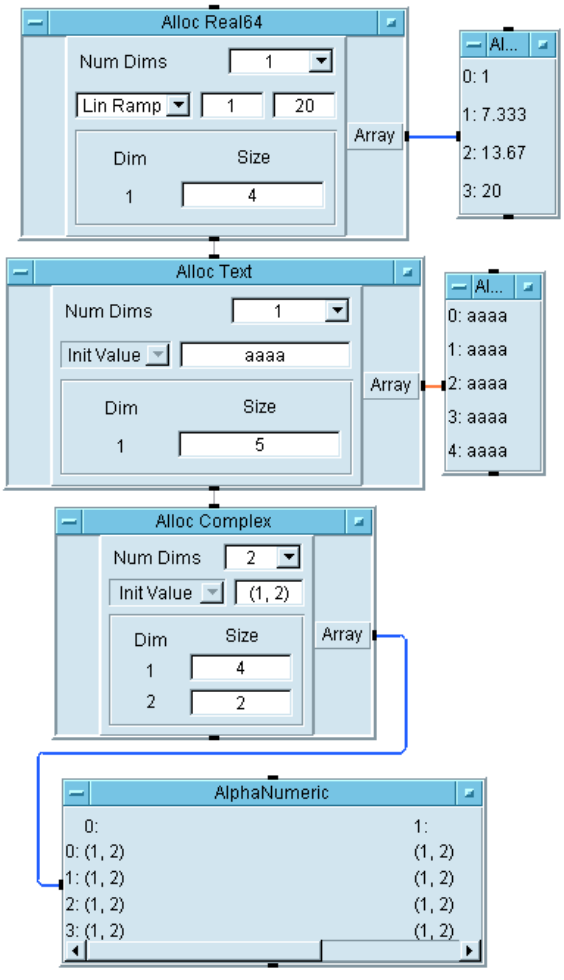
4.8.8 Razlaganje složenih podataka (*UnBuilde Data*)

Meni *UnBuilde Data* u sebi ima sedam objekata: *Coord, Complex, PComplex, Waveform, Arb Waveform, Spectrum* i *Record.* Ovi objekti se koriste za razlaganje na sastavne dijelove složenih podataka.

Primjer programa u programskom jeziku *VEE Pro* za korišćenje ovih objekata prikazan je na slici 4.42 i slici 4.43.

4.8.9 Pravljenje nizova (*Allocate Arrey*)

Meni *Allocate Arrey* u sebi ima devet objekata: *Alloc Text, Alloc UInt8, Alloc Int16, Alloc Int32, Alloc Real32, Alloc Real64, Alloc Coord, Alloc Complex i Allocovi PComplex.* Ovi objekti se koristite za pravljenje nizova (*Arrey*). Niz može biti jednodimenzioni ili višedimenzioni, a to zavisi od vrijednosti koja se upisuje u polje *Num Dims*. Dužina niza je određena vrijednost, koja se upisuje u polje *Size* (prvo polje predstavlja broj redova, a drugo broj kolona, kao u primjeru *Alloc Complex*). Mogu se praviti nizovi koji sadrže sve tipove podataka (tekst, razne brojeve, kordinate, kompleksne brojeve, itd.). U polje *Init Value* se upisuje vrijednost koja će se na početku upisati u sva polja niza. Kod brojnih nizova može se izabrati inicijalno popunjavanje niza i sa različitim elementim prema linearnoj ili logoritamskoj progresiji, u rasponu brojeva koji se mogu izabrati. Primjer programa u *VEE Pro* za korišćenje ovih objekata prikazan je na slici 4.44.



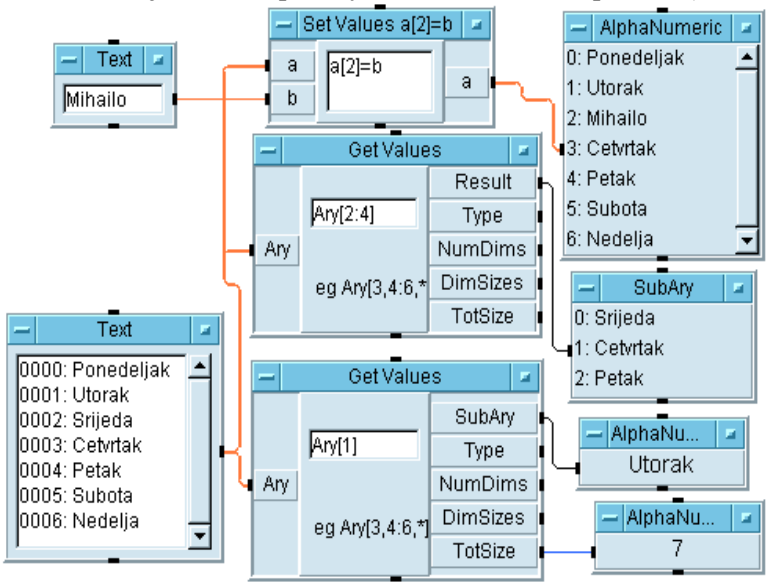
Slika 4.44 Objekti za pravljenje nizova

4.8.10 Pristup elementima niza (*Access Arrey*)

Meni *Access Array* u sebi ima četiri objekata: *Set Values, Get Values, Set Mappings* i *Get Mappings.* Ovi objekti se koristite za punjenje ili čitanje tačno određenog člana niza. Primjer programa u programskom jeziku *VEE Pro* za korišćenje ovih objekata prikazan je na slici 4.45.

Objekat ***Set Values*** se koristi za upis podataka u niz. Na jedan ulaz objekta se mora dovesti niz u koji se želi upisati nova vrijednost. Na drugi ulaz (može ih biti i više) se dovodi nova vrijednost, koja se želi upisati u niz. U formuli, koja se unosi u objekat potrebno je navesti redni broj člana niza u koji se upisuje nova vrijednost.

Objekat ***Get Values*** se koristi za čitanje podataka iz niza. Podaci se mogu čitati pojedinačno (tako što se u uglastu zagradu unese redni broj člana niza, koji se čita, npr. Ary[1] drugi član), ili odjednom više elemenata (tako što se u uglastu zagradu unese interval niza koji se čita, npr. Ary[2:4] treći, četvrti i peti član).



Slika 4.45 Objekti za rad sa nizovima

4.8.11 Rad sa tabelama (*Access Record*)

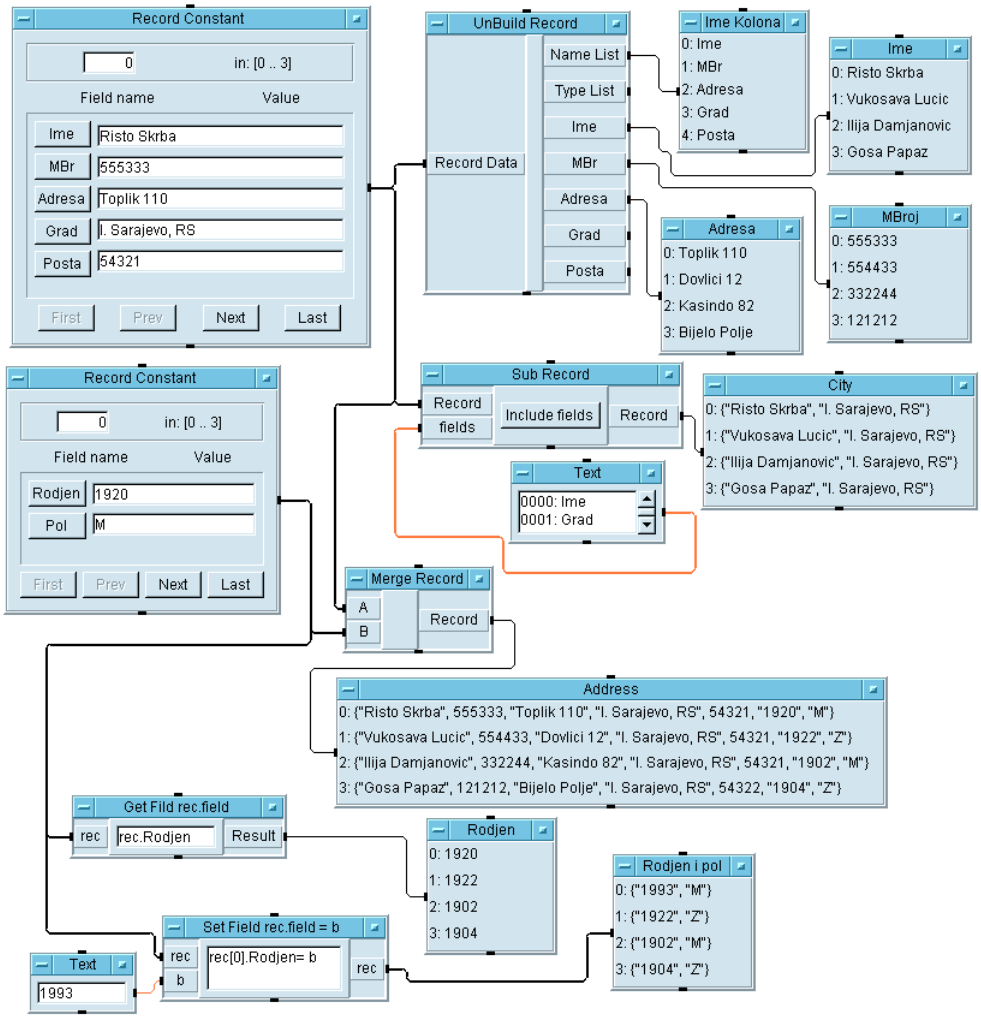
Meni *Access Record* u sebi ima četiri objekata: *Merge Record, Sub Record, Set Field* i *Get Field.* Ovi objekti se koristite za spajanje dva zapisa u jedan, izdvajanje samo određenih elemenata zapisa, upis novih vrijednosti u postojeći zapis ili uzimanje tačno određene vrijednosti zapisa. Zapis je najlakše shvatiti kao tabelu, gdje svaka kolona ima ime i predstavlja jedan tip podatka. Jedan red obuhvata različite podatke za svaku kolonu. Primjer programa u programskom jeziku *VEE Pro* za korišćenje ovih objekata prikazan je na slici 4.46.

Objekat ***Merge Record*** se koristi za spajanje dvije različite tabele (zapisa) u jednu tabelu, koja sadrži sve kolone i prve i druge tabele.

Objekat ***Sub Record*** se koristi da bi smo iz jedne tabele (zapisa) uzeli jednu ili više kolona i napravili novu tabelu. Na ulaz objekta pod nazivom *Fields* dovodi se ime kolone (ili lista od više kolona), koja se želi izdvojiti iz originalne tabele.

Objekat ***Set Field*** se koristi za upis podataka u tabelu (zapis). Mora se tačno navesti u koju kolonu (navođenjem imena kolone) i koji red (navođenjem rednog broja reda u uglastu zagradu) se želi upisati podatak.

Objekat ***Get Field*** se koristi za čitanje podataka iz tabele (zapisa). Mora se tačno navesti iz koje kolone (navođenjem imena kolone), se žele preuzeti podaci.



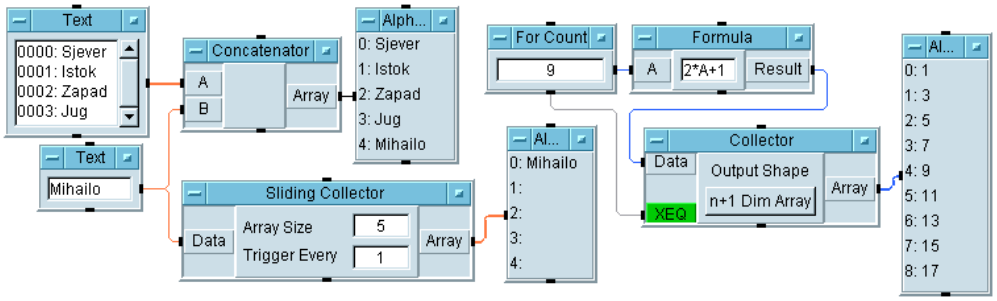
Slika 4.46 Objekti za rad sa zapisima

4.8.12 Spajanje nizova (Concatenator, Sliding Collector i Collector)

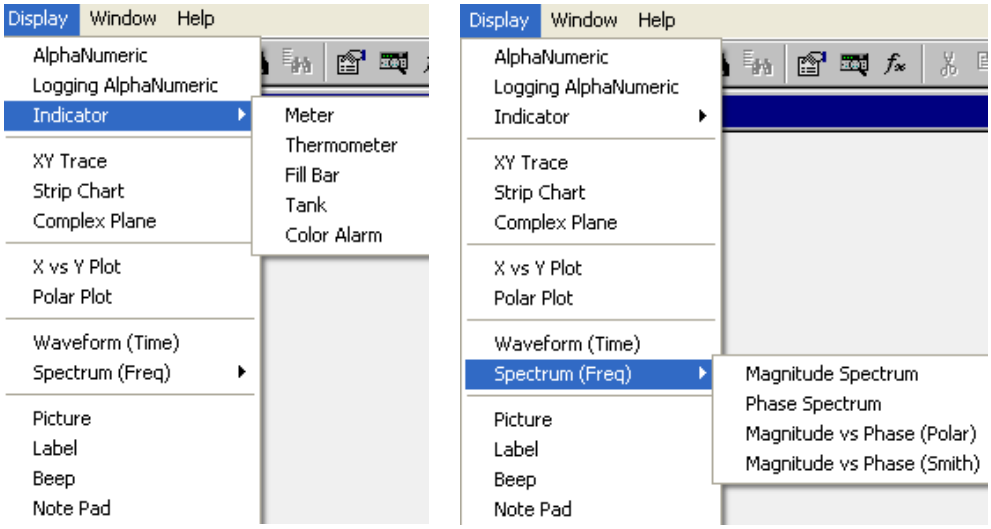
Objekat ***Concatenator*** se koristi za spajanje dva (ili više) podatka u jedan niz.

Objekat ***Sliding Collector*** se koristi za unos samo jednog podatka u niz, pri čemu se odmah definiše koliko će ukupno elemenata niz imati.

Objekat ***Collector*** se koristi za unos više podataka u niz. Podaci koji dolaze pojedinačno na ulaz objekta se pamte u njemu, ali biće upisani u niz tek, kada dođe kontrolni signal (sa vrijednošću 1) na ulaz XEQ. Ovaj objekat se obično kombinuje sa *For* petljom. Primjer programa u programskom jeziku *VEE Pro* za korišćenje ovih objekata prikazan je na slici 4.47.



Slika 4.47 Objekti za spajanje nizova



Slika 4.48 Meni Display (indikatori)

4.9 PRIKAZ PODATAKA (*DISPLAY*)

Osnovni elementi menija *Display* dati su na slici 4.48. U ovom meniju se nalaze objekti pomoću kojih se vrši prikazivanje rezultata. Programski jezik *VEE Pro* ima

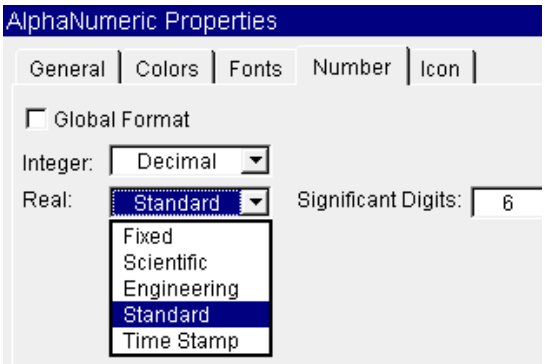
široku lepezu indikatora od jednostavnog digitalnog displeja, preko simulacije analognih instrumenata, pa sve do raznoraznih XY indikatora. Posebno treba naglasiti više indikatora, koji omogućuju prikaz spektra AC signala.

4.9.1 Digitalni displej (*AlphaNumjeric i Logging AlphaNumjeric*)

Objekat ***AlphaNumjeric*** se koristi za prikaz skalarnih vrijednosti, jednodimenzionih i dvodimenzionih nizova (kada se vrijednosti prikazuju u dvije kolone). Ako nema dovolno mjesta za prikaz niza, onda se u ovom objektu automatski pojavljuje ruler. Ovo je jedan od objekata koji se najviše koristi u programskom jeziku *VEE Pro* za prikaz brojeva, ali se koristiti i za prikaz teksta.

Objekat ***Logging AlphaNumjeric*** se koristi za prikaz skalarnih vrijednosti i jednodimenzionih nizova.

Number je osobina, koja se najviše podešava kod ova dva indikatora, vidi sliku 4.49. Da bi se mogla podešavati ova osobina opcija *Global Format* treba biti ne čekirana. Kod parametra *Real* se bira način prikaza brojeva. Ako se izabere opcija *Time Stamp,* tada ovaj objekat prikazuje datum ili vrijeme. Pomoću parametra *Significant Digits* se kod realnih brojeva podešava broj cifara koje će biti prikazane iza decimalne zapete.



Slika 4.49 Osobina Number

4.9.2 Analogni prikaz podataka (*Indicator*)

Meni *Indicator* u sebi ima pet objekata: *Meter, Thermometer, Fill Bar, Tank* i *Color Alarm.* Ovi objekti se koristite za razne vrste analognih prikaza.

Za svaki indikator se bira početak i kraj mjernog opsega. Skala instrumenta može biti linearna ili logoritamska. Svaka skala indikatora se može obojiti u tri boje: zelenu, žutu i crvenu, u zavisnosti od brojne vrijednosti, koja dođe na ulaz ovog objekta. Ovi indikatori se koriste u slučajevima, kada se neki signal brzo mijenja, odnosno kada je pokazivanje digitalnog indikatora nestabilno i nije moguće izvršiti tačno očitanje. U ovim slučajevima analogni indikator je pogodniji jer se može lako pratiti brzina i pravac promjene neke vrijednosti. Primjer za ovo je brzinomjer kod automobila. Primjer programa u programskom jeziku *VEE Pro* za korišćenje ovih objekata prikazan je na slici 4.50.

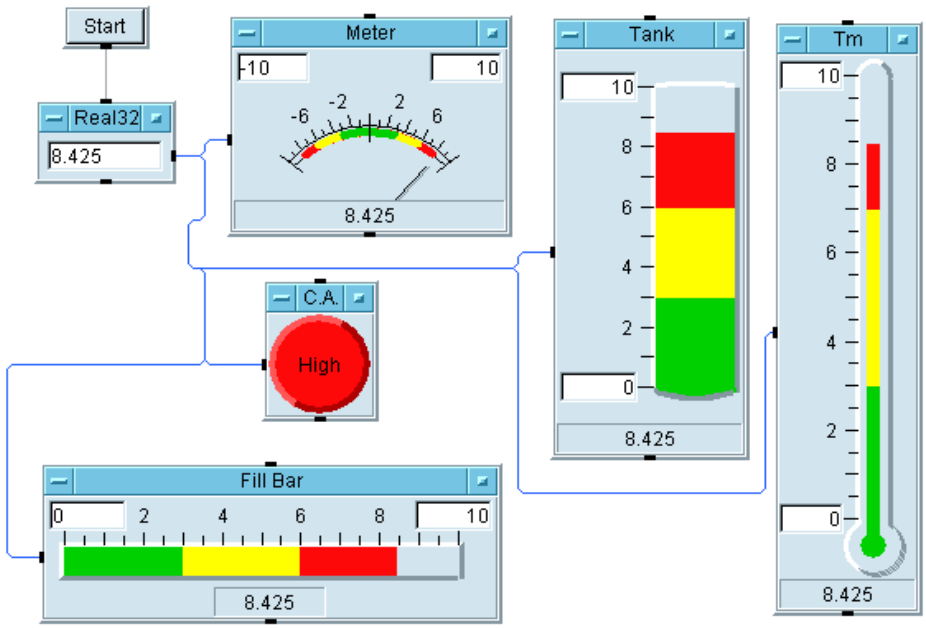
Objekat ***Meter*** se koristi za prikaz brojnih vrijednosti na indikatoru, koji predstavlja simulaciji analognog instrumenta. Na istom objektu se može izvršiti i prikaz digitalne vrijednosti istog signala.

Objekat ***Thermometer*** se koristi za prikaz brojnih vrijednosti na indikatoru, koji predstavlja simulaciji vertikalnog (ili horizontalnog) kapilarnog termometra. Na istom objektu se može izvršiti i prikaz digitalne vrijednosti.

Objekat ***Fill Bar*** se koristi za prikaz brojnih vrijednosti na indikatoru, koji predstavlja simulaciji vertikalnog (ili horizontalnog) bar grafa. Na istom objektu se može izvršiti i prikaz digitalne vrijednosti.

Objekat ***Tank*** se koristi za prikaz brojnih vrijednosti na indikatoru, koji predstavlja simulaciji vertikalnog rezervoara.

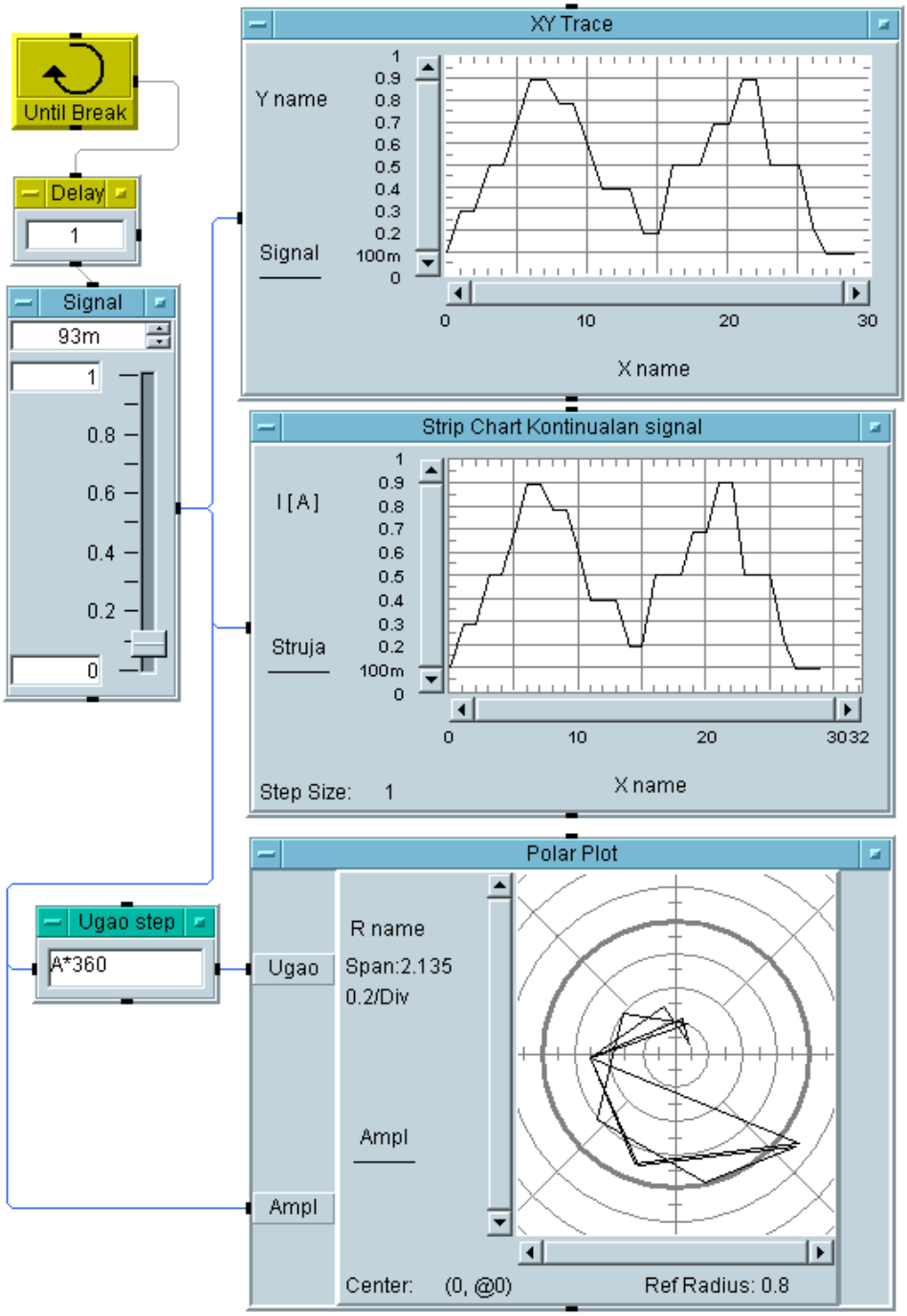
Objekat ***Color Alarm*** koristi se najčešće za signalizacije i alarme. Čini ga krug koji može biti obojen u jednu od tri boje i na kome mogu pisati tri različita teksta (inicijalno *Low, Mid* i *High*). Boja kruga i teksta koji se pojavljuje, zavisi od trenutne brojne vrijednosti, koja dolazi na ulaz objekta.



Slika 4.50 Razni indikatori

4.9.3 Grafički prikaz podataka (*XY Trace, Strip Chart i Polar Plot*)

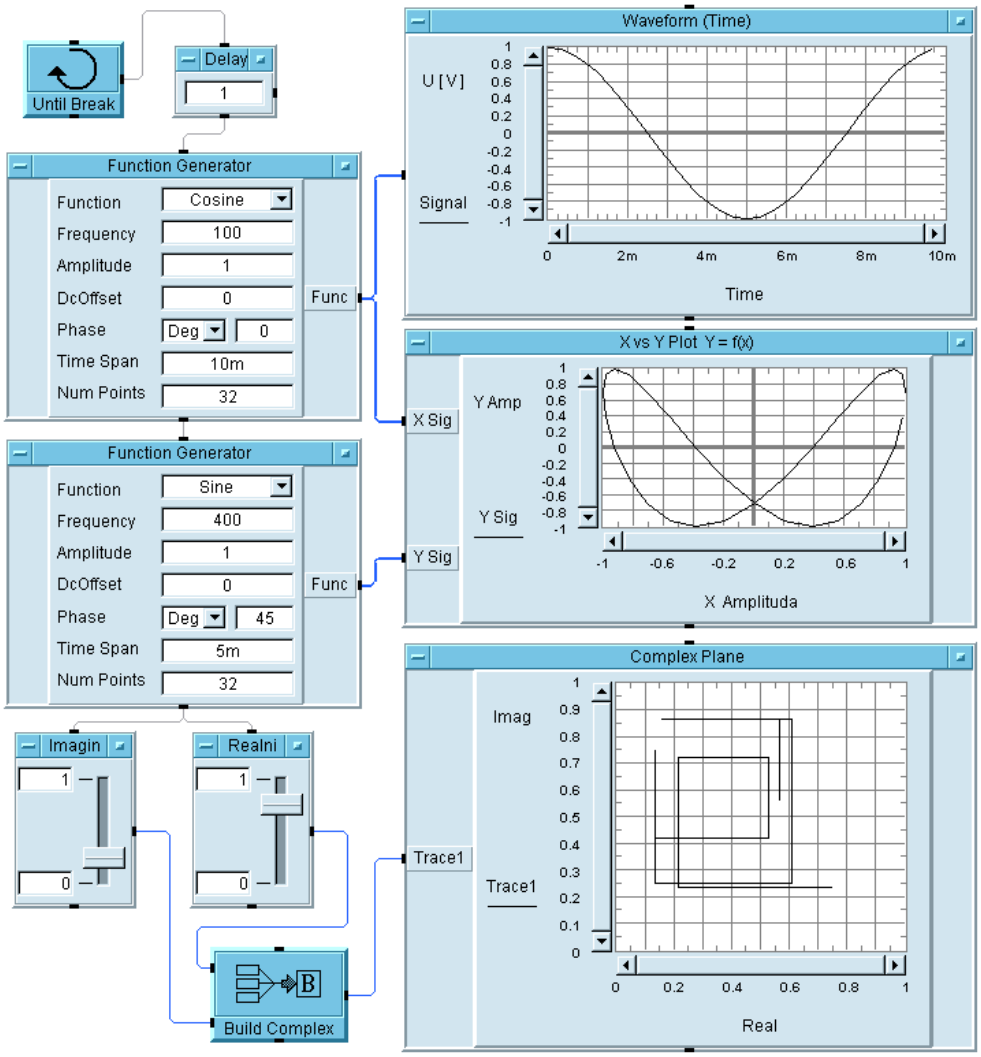
Objekat ***XY Trace*** se koristi za prikaz vremenski promjenljivog signala. Na Y osi se prikazuje amplituda signala, a na X osi se prikazuje vrijeme u sekundama kada signal dolazi. Nema vremenskog ograničenja za snimanje na ovom objektu, a vremenska osa se automatski mijenja, kako se mijenja vrijeme snimanja signala.



Slika 4.51 XY, skript i polarni dijagram

Objekat ***Strip Chart*** se koristi za prikaz vremenski promjenljivog signala. Na Y osi se prikazuje amplituda signala, a na X osi se prikazuje redni broj signala koji je došao na ulaz dijagrama. Ovaj dijagram će pamtiti i prikazivati samo ulazni signala do određenog podešenog broja odbiraka (parametar *Buffer Size*).

Objekat ***Polar Plot*** se koristi za prikaz signala u polarnom kordinatnom sistemu. Na jedan ulaz ovog indikatora se dovodi amplituda signala, a na drugi ulaz se dovodi faza signala u stepenima. Na indikatoru se prikazuje svaka vrijednost ulaznog signala (kombinacija amplitude i faze), a svake dvije susjedne tačke na indikatoru se spajaju linijom. Primjer korišćenje ovih objekata dat je na slici 4.51.



Slika 4.52 Vremenski, Y = f(x) i kompleksni dijagram

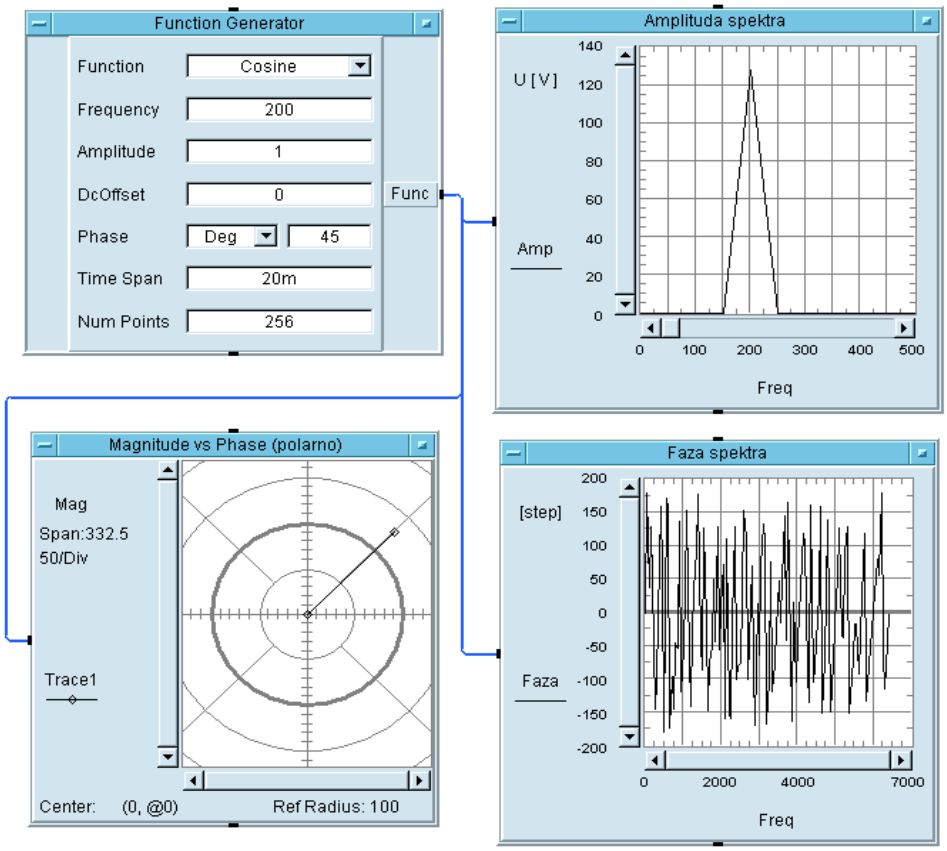
4.9.4 Osciloskop (*X vs Y Plot, Complex Plane i Waveform Time*)

Objekat ***X vs Y Plot*** se koristi za prikaz međusobne zavisnosti dva signala, koji se dovode na dva ulaza ovog objekta. Ovaj indikator se najčešće koristi kao simulacija osciloskopa, za pravljenje Lisažeovih figura, kada se na ulaz dovode naizmjenični signali različitih frekvencija.

Objekat ***Complex Plane*** se koristi za prikaz kompleksnih signala, koji moraju imati dvije kordinate, realnu i imaginarnu. Na Y osi se prikazuje amplituda imaginarne komponente, a na X osi se prikazuje amplituda realne komponente. Svake dvije susjedne tačke na indikatoru se spajaju linijom.

Objekat ***Waveform (Time)*** se koristi za prikaz signala u vremenskom domenu. Na Y osi se prikazuje amplituda signala, a na X osi se prikazuje vrijeme u sekundama (ili mili sekundama), kada je signal dolazio na ulaz dijagrama. Nema vremenskog ograničenja za snimanje na ovom objektu, a vremenska osa se automatski mijenja, kako se mijenja vrijeme snimanja signala.

Primjer korišćenje ovih objekata dat je na slici 4.52.



Slika 4.53 Frekventni dijagrami

4.9.5 Frekventna analiza naizmjeničnih signala (*Spectrum Freq*)

Meni *Spectrum (Freq)* u sebi ima četiri objekata: *Magnitude Spectrum, Phase Spectrum, Magnitude vs Phase (Polar)* i *Magnitude vs Phase (Smith)*. Ovi objekti se koristite za razne vrste frekventnih analiza naizmjeničnih signala.

Objekat ***Magnitude Spectrum*** se koristi za prikaz amplitudno frekventne karakteristike kod naizmjeničnih signala. Na Y osi ovog dijagrama se prikazuje amplituda, a na X osi frekvencija signala. Pik, odnosno maksimum signala na dobijenom dijagramu se pojavljuje samo na osnovnoj frekvenciji ulaznog signala.

Objekat ***Phase Spectrum*** se koristi za prikaz fazno frekventne karakteristike kod naizmjeničnih signala. Na Y osi ovog dijagrama se prikazuje faza, a na X osi frekvencija signala.

Objekti ***Magnitude vs Phase (Polar)*** i ***Magnitude vs Phase (Smith)*** se koriste za prikaz amplitudno fazne karakteristike naizmjeničnih signala u polarnom kordinatnom sistemu. Primjer korišćenja ovih objekata dat je na slici 4.53.

4.9.6 Slika, tekst, zvuk i komentari (*Picture, Label, Beep i Note Pad*)

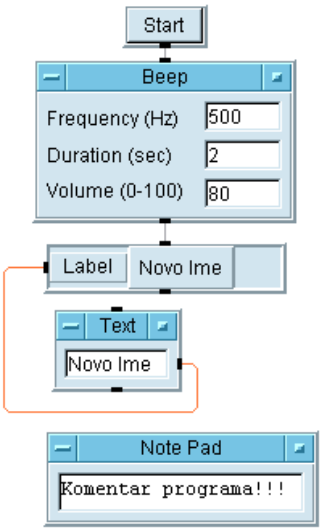
Objekat ***Picture*** se koristi za prikaz slike na radnoj površini za pisanje programa.

Objekat ***Label*** se koristi za prikaz teksta u programu. Obično je to neki kratak tekst do par riječi, koji predstavlja obično naziv nekog objekta. Tekst u ovom objektu se može mijenjati i programski preko kontrolnog ulaza *Label*.

Objekat ***Beep*** se koristi za generisanje zvuka u programu, obično kao znak upozorenja. Frekvencija zvuka se mijenja upisom brojne vrijednosti u polje *Frequency (Hz)*. Može se mijenjati dužina trajanja i jačina zvuka.

Objekat ***Note Pad*** se koristi za prikaz teksta, koji predstavlja komentar za napisani program. U ovaj objekat se može upisati tekst od nekoliko redova. Ovaj objekat stoji samostalno u programu i ne povezuje se sa drugim objektima.

Primjer korišćenja ovih objekata dat je na slici 4.54.

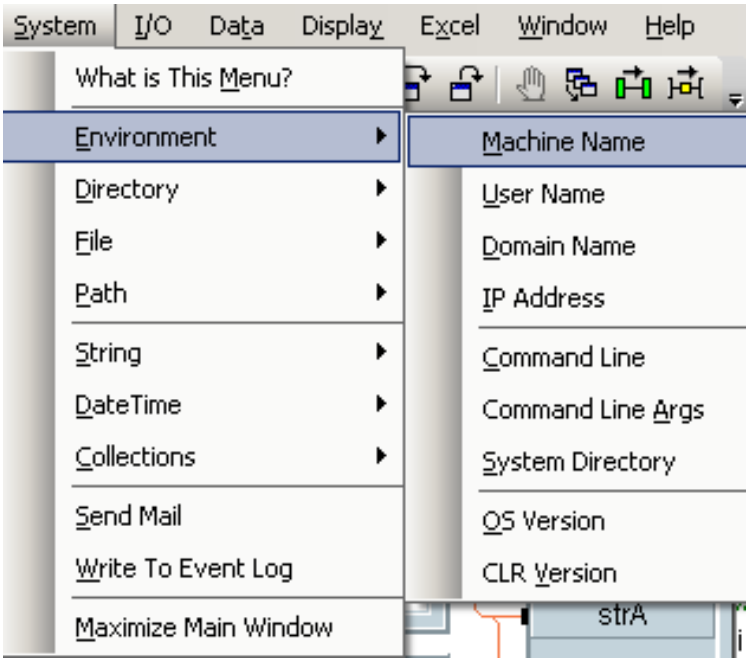


Slika 4.54 Objekat Beep

4.10 PRISTUP OPERATIVNOM SISTEMU RAČUNARA (*SYSTEM*)

Osnovni elementi menija *System* dati su na slici 4.49. U ovom meniju se nalaze funkcije koje omogućuje *.NET Framework Class Library (FCL)*, u verziji programa *VEE Pro* 8.5 i novijim. Ove funkcije omogućuju jednostavan pristup

funkcijama operativnog sistema računara na kome je program instalisan. Postoji dosta primjera programa koji demonstriraju ove funkcije. Oni se mogu naći u *Helpu* preko *Help menu* → *Open Example...* → *DotNET* direktorijum.



Slika 4.49 Meni System

4.10.1 Pristup hardveru i softveru računara (*Envirnoment*)

Funkcije *Envirnoment* se koriste za pronalaženje osnovnih hardverskih i softverskih svojstava računara na kome je program *VEE Pro* instaliran.

Funkcija ***Machine Name*** se koristi za prikaz imena računara.

Funkcija ***User Name*** se koristi za prikaz imena trenutnog korisnika računara.

Funkcija ***Domain* Name** se koristi za prikaz imena domena čiji je korisnik član.

Funkcija ***IP Address*** se koristi za prikaz IP adrese računara.

Funkcija ***Command Line*** se koristi za prikaz putanje do direktorijuma na kome se nalazi izvršna verzija programa *VEE Pro*, odnosno fajl ***vee.exe***.

Funkcija ***Comand Line Args*** se koristi za pretvaranje putanje do direktorijuma na kome se nalazi izvršna verzija programa *VEE Pro* (fajl ***vee.exe***), u jednodimenzioni tekstualni niz.

Funkcija ***System Directory*** se koristi za prikaz putanje do direktorijuma na kome se nalazi instalisan operativni sistem računara.

Funkcija ***OS Version*** se koristi za prikaz imena i verzije instalisanog operativnog sistema.

Funkcija ***CLR Version*** se koristi za prikaz verzije *Common Language Runtime (CLR)*, koja je instalisana na računaru.

4.10.2 Rad sa direktorijumima (*Directory*)

Funkcije *Directory* se koriste za rad sa direktorijumima računara na kome se trenutno izvršava program *VEE Pro*.

Funkcija ***Create Directory*** se koristi za pravljenje novog direktorijuma.

Funkcija ***Delete Directory*** se koristi za brisanje već postojećeg direktorijuma.

Funkcija ***Directory Exist?*** se koristi za ispitivanje da li postoji navedeni direktorijum bilo gdje na računaru.

Funkcija ***Get Files*** se koristi za prikaz svih fajlova na zadanom direktorijumu, u vidu jednodimenzionog tekstualnog niza.

Funkcija ***Get Specifed Files*** se koristi za prikaz, u vidu niza, svih fajlova sa zadanom ekstenzijom (na primjer, *.vee, *.exe, *.doc, *.* ...), koji se nalaze na specificiranom direktorijumu.

Funkcija ***Get Subdirectories*** se koristi za prikaz svih direktorijuma, koji se nalaze na specificiranom direktorijumu.

Funkcija ***Get Current Directory*** se koristi za prikaz direktorijuma na kome je trenutno instalisan kompletan program *VEE Pro* sa svim pratećim dokumentima i direktorijima.

Funkcija ***Get Parent Directory*** se koristi za prikaz putanje do predhodnog direktorijuma na kome se nalazi trenutno specificiran direktorijum.

Funkcija ***Get Root Directory*** se koristi za prikaz korijena (C:, D: ili neki drugi disk) trenutno specificiranog direktorijuma.

Funkcija ***Get Logical Drivers*** se koristi za prikaz naziva svih particija hard diska i svih drugih trenutno aktivnih spoljnih diskova.

Funkcija ***Last Directory Write Time*** se koristi za prikaz vremena, kada je zadnji put vršena modifikavija na specificiranom direktorijumu.

Funkcija ***Last Directory Access Time*** se koristi za prikaz vremena, kada je zadnji put vršen pristup specificiranom direktorijumu.

4.10.3 Rad sa dokumentima (*File*)

Funkcije *File* se koriste za rad sa dokumentima, koji već postoje na računaru, kao i za pronalaženje osnovnih podataka vezanih za vrijeme nastanka i promjena na dokumentima.

Funkcija ***Delete File*** se koristi za brisanje svih dokumenata, koji se nalaze na specificiranom direktorijumu.

Funkcija ***Move File*** se koristi za premještanje dokumenta sa jedne lokacije na drugu. Može se koristiti i za promjenu imena dokumenta.

Funkcija ***File Exist?*** se koristi za ispitivanje da li postoji navedeni fajl na specificiranom direktorijumu.

Funkcija ***Get File Creation Time*** se koristi za prikaz vremena, kada je navedeni fajl snimljen na specificiranom direktorijumu.

Funkcija ***Get Last File Access Time*** se koristi za prikaz vremena, kada se zadnji put pristupilo navedenom fajl snimljen na specificiranom direktorijumu.

Funkcija ***Get Last File Write Time*** se koristi za prikaz vremena, kada je navedeni fajl nastao.

Funkcija ***Set File Creation Time*** se koristi za promjenu vrijednosti vremena, kada je fajl snimljen na računar.

Funkcija ***Set Last File Access Time*** se koristi za promjenu vrijednosti vremena, kada se zadnji put pristupilo fajlu.

Funkcija ***Set Last File Write Time*** se koristi za promjenu vrijednosti vremena, kada je fajl nastao.

4.10.4 Rad sa imenom i ekstenzijom dokumenata (*Path*)

Funkcije *Path* se koriste za rad sa dokumentima, koji već postoje na računaru, kao i obrade podataka koje su vezane za ime i ekstenziju dokumenata.

Funkcija ***Get Directory Name*** se koristi za pronalaženje imena direktorijuma na kome se nalazi uneseni dokument.

Funkcija ***Get File Name*** se koristi za pronalaženje na unesenom direktorijumu imena dokumenta sa ekstenzijom.

Funkcija ***Get File Name Without Extension*** se koristi za pronalaženje na unesenom direktorijumu imena dokumenta bez ekstenzije.

Funkcija ***Get Ful Path*** se koristi za pronalaženje imena dokumenta i putanje na disku do njega na unesenoj lokaciji.

Funkcija ***Get Path Root*** se koristi za pronalaženje korijena direktorijuma na kome se nalazi unesena putanja.

Funkcija ***Is Path Rooted?*** se koristi za provjeravanje da li se određeni dio putanje nalazi u kompletnoj putanje do određenog dokumenta.

Funkcija ***Has Extension?*** se koristi za provjeravanje da li na unesenom direktorijumu postoji zadata ekstenzija dokumenta.

Funkcija ***Get Extension*** se koristi za izdvajanje ekstenzije dokumenta iz punog naziva dokumenta.

Funkcija ***Change Exstension*** se koristi za promjenu postojeće ekstenzije dokumenta.

Funkcija ***Get Temp File Name*** se koristi za pravljenje *temporary* fajla na lokaciji *C:\Documents and Settings\{UserName}\Local Settings\Temp.*

Funkcija ***Get Temp Directory Path*** se koristi za pravljenje *temporary* direktorijuma na lokaciji *C:\Documents and Settings\{UserName}\Local Settings\Temp.*

4.10.5 Rad sa tekstom (*String*)

Funkcije *String* se koriste za rad sa tekstualnim podacima, koji mogu da se porede, mijenjaju i pretražuju na razne načine.

Funkcija ***Compare*** se koristi za poređenje dva stringa.

Funkcija ***Remove*** se koristi za uklanjanje jednog teksta iz drugog.

Funkcija ***Replace*** se koristi za automatsku zamjenu jednog ili više karaktera u kompletnom tekstu.

Funkcija ***Split*** se koristi za podjelu teksta na više dijelova. Potrebno je da se unese jedan ili više karaktera, koji služe kao okidači dijeljenja. Od ulaznog teksta se dobije niz.

Funkcija ***Join*** se koristi da bi od jednodimenzionog niza dobili jedinstven tekst. Pri tome se može izabrati separator, koji će u novom tekstu razdvajati elemente niza, a za to se najčešće koristi ", ".

Funkcija ***Start With*** se koristi za provjeru da li neki tekst počinje sa nekim slovom ili grupom slova. Kao rezultat se dobije logička promjenljiva, koja može imati vrijednost *True* ili *False*.

Funkcija ***Ends With*** se koristi za provjeru da li neki tekst završava sa nekim slovom ili grupom slova. Kao rezultat se dobije logička promjenljiva, koja može imati vrijednost *True* ili *False*.

Funkcija ***Pad Left*** se koristi za dodavanje unesenog karaktera sa lijeve strane ulaznog teksta. Koliko karaktera se dodaje zavisi od broja koji se unosi na ulaz funkcije pod nazivom *TotalWith*. Ovaj broj označava maksimalni broj karaktera koliko konačan tekst može da ima.

Funkcija ***Pad Right*** se koristi za dodavanje unesenog karaktera sa desne strane ulaznog teksta. Koliko karaktera se dodaje zavisi od broja koji se unosi na ulaz funkcije pod nazivom *TotalWith*. Ovaj broj označava maksimalni broj karaktera koliko konačan tekst može da ima.

4.10.6 Rad sa datumom i vremenom (*DateTime*)

Funkcije *DateTime* se koriste za rad sa podacima, koji predstavljaju datum i vrijeme. Kao osnova za rad sa svim ovim funkcijama se koristi sistemski datum i vrijeme sa računara. Ove funkcije se mogu koristiti i za obradu datuma, koji se dobiju kao rezultat nekih drugih funkcija.

Funkcija ***Now*** se koristi za prikazivanje sistemskog datuma i vremena, koji su postavljeni na računaru.

Funkcija ***UtcNow*** se koristi za prikazivanje *Universal Time (Greenwich Mean Time)* sistemskog datuma i vremena, koji su postavljeni na računaru, ali u odnosu na vrijeme koje je u tom trenutku po Griniču.

Funkcija ***Add Seconds*** se koristi za pomjeranje sistemskog vremena, ili bilo kog drugog vremena, koje se dovodi na ulaz ove funkcije, za određeni broj sekundi.

Funkcija ***Add Minutes*** se koristi za pomjeranje sistemskog vremena, ili bilo kog drugog vremena, koje se dovodi na ulaz ove funkcije, za određeni broj minuta.

Funkcija ***Add Hours*** se koristi za pomjeranje sistemskog vremena, ili bilo kog drugog vremena, koje se dovodi na ulaz ove funkcije, za određeni broj sati.

Funkcija ***Add Days*** se koristi za pomjeranje sistemskog datuma, ili bilo kog drugog datuma, koji se dovodi na ulaz ove funkcije, za određeni broj dana.

Funkcija ***Add Months*** se koristi za pomjeranje sistemskog datuma, ili bilo kog drugog datuma, koji se dovodi na ulaz ove funkcije, za određeni broj dana.

Funkcija ***Add Years*** se koristi za pomjeranje sistemskog datuma, ili bilo kog drugog datuma, koji se dovodi na ulaz ove funkcije, za određeni broj godina.

Funkcija ***Subtract*** se koristi za izračunavanje razlike između dva datuma odnosno vremena. Razlika se prikazuje u obliku: broj dana, sati:minuta:sekundi.

Funkcija ***Compare*** se koristi za poređenje dva datuma, kako bi se utvrdilo koji datum je stariji. Ako je datum, koji se dovodi na ulaz *T1* stariji od datuma, koji se dovodina ulaz *T2* rezultat je "-1". Ako je datum, koji se dovodi na ulaz *T2* stariji od datuma, koji se dovodina ulaz *T1* rezultat je "1". Ako su datumi isti rezultat je "0".

Funkcija ***To String*** se koristi za prikazivanje sistemskog datuma i vremena u formatu podataka string (tekst).

Funkcija ***To Local Time*** se koristi za pretvaranje UTC vremena, u lokalno vrijeme podešeno na računaru.

Funkcija ***To Universal Time*** se koristi za pretvaranje lokalnog vremena, koje je podešeno na računaru u UTC vrijeme.

Funkcija ***Day of Week*** se koristi za prikaz imena dana u sedmici za datum, koji je doveden na ulaz ove funkcije.

Funkcija ***Day of Year*** se koristi za prikaz broja dana u godini za datum, koji je doveden na ulaz ove funkcije.

Funkcija ***Is Leap Year?*** se koristi za utvrđivanje da li je godina dovedena na ulaz ove funkcije prestupna (izlaz je *True*) ili nije (izlaz je *False*).

5 POVEZIVANJE SA DRUGIM PROGRAMIMA

Jedna od dobrih osobina programskog jezika *VEE Pro* je mogućnost direktnog povezivanja sa dokumentima napravljenim u drugim programima, pa čak i kreiranje potpuno novih dokumenata u tim programima. *VEE Pro* najčešće komunicira sa dokumentima napravljenim u *Microsoft Excelu*, tako što u njih upisuje rezultate mjerenja ili iz njih preuzima neke predhodno snimljene podatke. Može se ostvariti i direktna komunikacija sa bazom podataka napravljenom u *Microsoft Accessu*, kako bi se u bazu direktno upisivali rezultati mjerenja ili iz baze preuzimali ranije snimljeni podaci. Rjeđe se koristi povezivanje sa tekstualnim dokumentima napravljenim u *Microsoft Wordu*, a češće se brojni i tekstualni podaci upisuju direktno u tekstualni fajl, da bi se zauzelo što manje memorijskog prostora. U ovom poglavlju biće predstavljeno kako se program napisan u programskom jeziku *VEE Pro* može direktno pozivati i koristiti iz programa napisanog u programskom jeziku *Visual Basic*.

5.1 *MICROSOFT EXCEL*

Programski jezik *VEE Pro* ima mogućnost pravljenja programa, koji direktno komunicira sa dokumentima napravljenim u *Microsoft Excelu*, ali samo u verziji Excel97, kako bi se ubrzala komunikacija između ova dva programa. *VEE Pro* najčešće komunicira sa dokumentima napravljenim u *Microsoft Excelu*, tako što u njih upisuje rezultate mjerenja ili iz njih preuzima neke predhodno snimljene podatke. Da bi se ova komunikacija ostvarila potrebno je prvo deklarisati promjenljive. Primjer deklarisanja promjenljivih prikazan je na slici 5.1. U ovom primjeru deklarisane su sljedeće promjenljive:

- *ExcelApp*
- *xlsFileName*
- *xlsFileSpec*
- *ExcelWkbk*
- *Wks*
- *range*.

Promjenljiva ***ExcelApp*** namijenjena je za komunikaciju sa kompletnim *Excel* programom. Mora biti deklarisana na globalnom nivou kako bi se mogla koristiti u glavnom programu, ali i u svim podprogramima i funkcijama. Tip podatka treba deklarisati kao *Variant*, što znači da se preko ove promjenljive mogu slati i primati svi tipovi podataka.

Promjenljiva ***xlsFileName*** namijenjena je za rad sa imenom *Excel* dokumenta. Koristi se prilikom otvaranja već postojećeg dokumenta, ili prilikom spašavanja

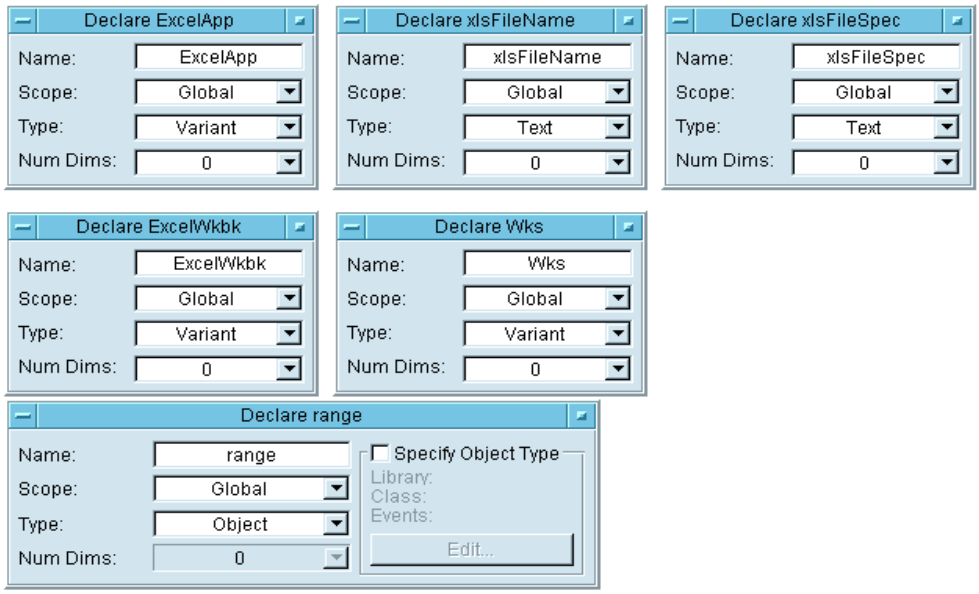
dokumenta pod novim imenom. Mora biti deklarisana na globalnom nivou. Tip podatka treba deklarisati kao *Text*.

Promjenljiva ***xlsFileSpec*** namijenjena je za rad sa putanjom, odnosno direktorijumom na kome se nalazi *Excel* dokument. Koristi se prilikom otvaranja već postojećeg dokumenta, ili spašavanja dokumenta pod novim imenom. Mora biti deklarisana na globalnom nivou. Tip podatka treba deklarisati kao *Text*.

Promjenljiva ***ExcelWkbk*** namijenjena je za rad sa knjigom (*Book*), koja čini *Excel* dokument. Koristi se prilikom otvaranja novog dokumenta. Mora biti deklarisana na globalnom nivou. Tip podatka treba deklarisati kao *Variant*.

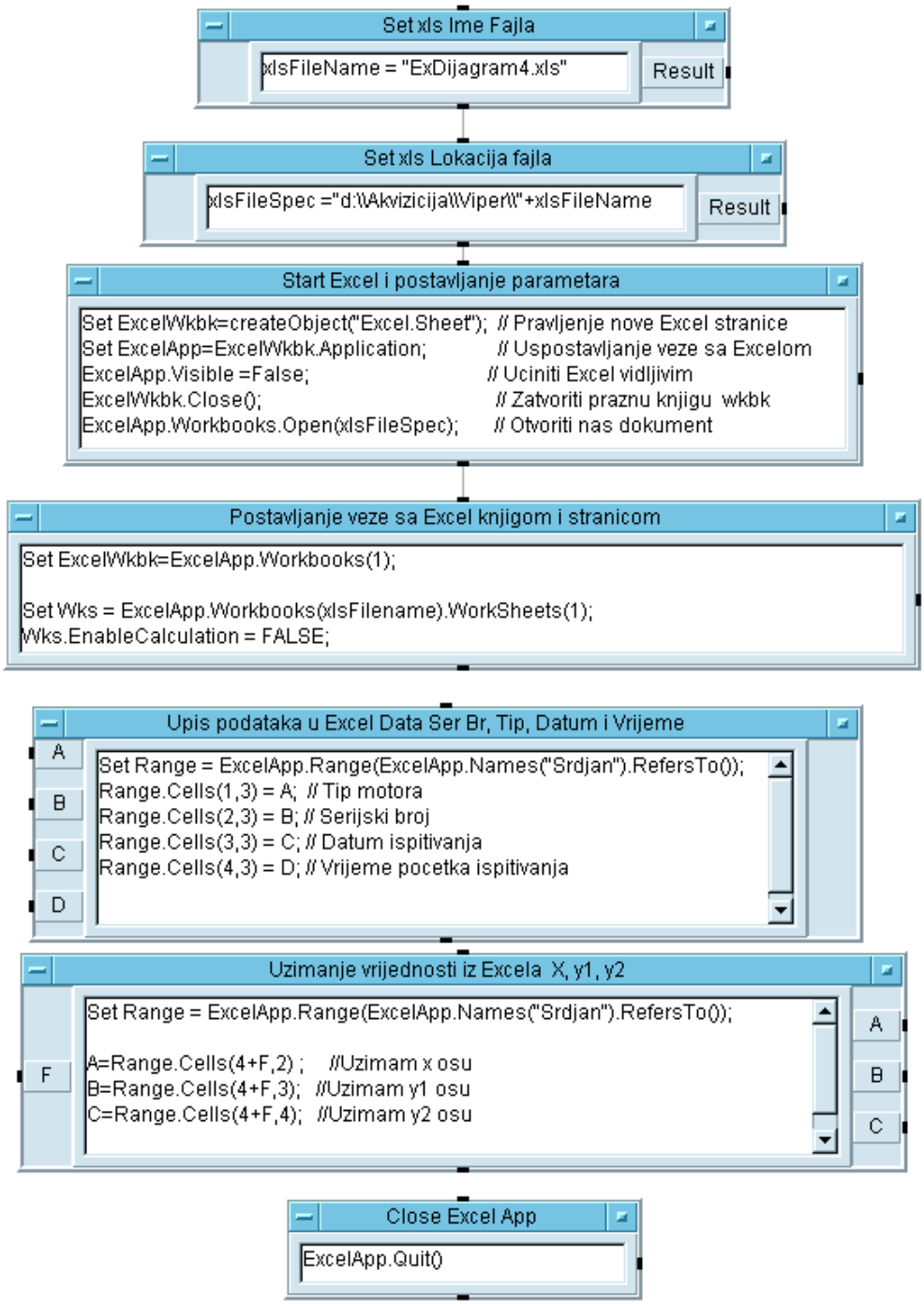
Promjenljiva ***Wks*** namijenjena je za rad sa konkretnom stranicom (*Sheets*), koja čini *Excel* dokument. Koristi se prilikom otvaranja novog dokumenta. Mora biti deklarisana na globalnom nivou. Tip podatka treba deklarisati kao *Variant*.

Promjenljiva ***range*** namijenjena je za definisanje početne ćelije u *Excel* dokumentu od koje su dalje definisane sve ostale ćelije na toj stranici. Koristi se prilikom upisa i čitanja podataka. Mora biti deklarisana na globalnom nivou. Tip podatka treba deklarisati kao *Object*.



Slika 5.1 Deklarisanje promjenljivih za komunikaciju sa Excelom

Na slici 5.2 prikazan je dio programa, koji se koristi za upis i čitanje podataka iz već postojećeg *Excel* dokumenta "ExDijagram4.xls" koji se nalazi na direktorijumu "d:\\Akvizicija\\Viper\\". Prilikom upisa i čitanja podataka iz neke ćelije navodi se prvo redni broj kolone, a zatim redni broj reda (npr. *Cell*(2,3) druga kolona, treći red) u odnosu na referentnu ćeliju. Na kraju rada sa *Excel* dokumentima potrebno je zatvoriti Excel aplikaciju, a to se radi sa komandom "*ExcelApp.Quit()*".



Slika 5.2 Primjer programa za upis i čitanje podataka iz Excela

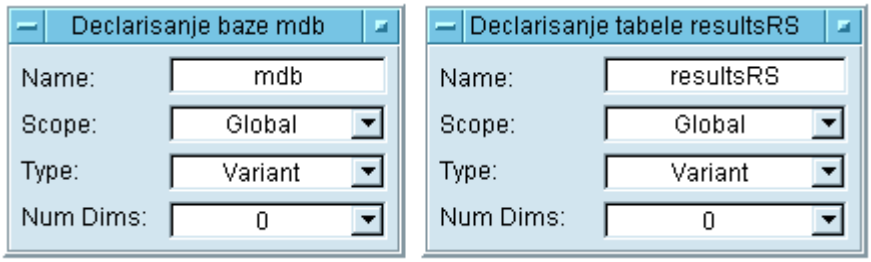
5.2 *MICROSOFT ACCESS*

Programski jezik *VEE Pro* ima mogućnost pravljenja programa, koji direktno komunicira sa bazom podataka napravljenom u *Microsoft Accessu,* ali samo u verziji *Access97,* kako bi se ubrzala komunikacija između ova dva programa. Program napisan u *VEE Pro* najčešće komunicira sa bazom, tako što u nju upisuje rezultate mjerenja ili iz baze preuzima neke predhodno snimljene podatke. Da bi se ova komunikacija ostvarila potrebno je prvo deklarisati promjenljive. Primjer deklarisanja promjenljivih prikazan je na slici 5.3. U ovom primjeru deklarisane su sljedeće promjenljive:

- *mdb*
- *resultsRS*.

Promjenljiva ***mdb*** namijenjena je za komunikaciju sa kompletnim *Access* programom. Mora biti deklarisana na globalnom nivou kako bi se mogla koristiti u glavnom programu, ali i svim podprogramima i funkcijama. Tip podatka treba deklarisati kao *Variant*, što znači da se preko ove promjenljive mogu slati i primati svi tipovi podataka.

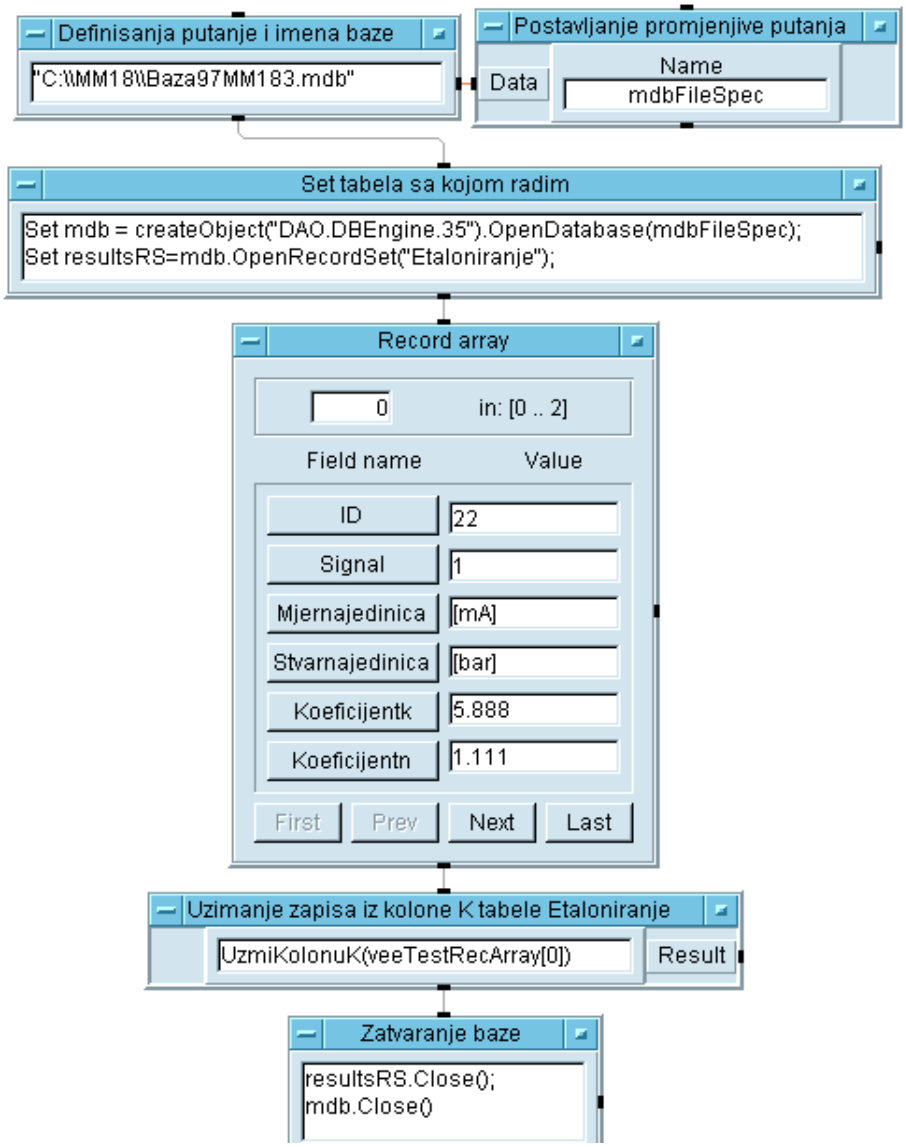
Promjenljiva ***resultsRS*** namijenjena je za komunikaciju sa jednom tabelom u bazi. Ako baza ima više tabela, onda program mora imati posebno deklarisanu promjenljivu za svaku tabelu. Ova promjenljiva mora biti deklarisana na globalnom nivou. Tip podatka treba deklarisati kao *Variant*.



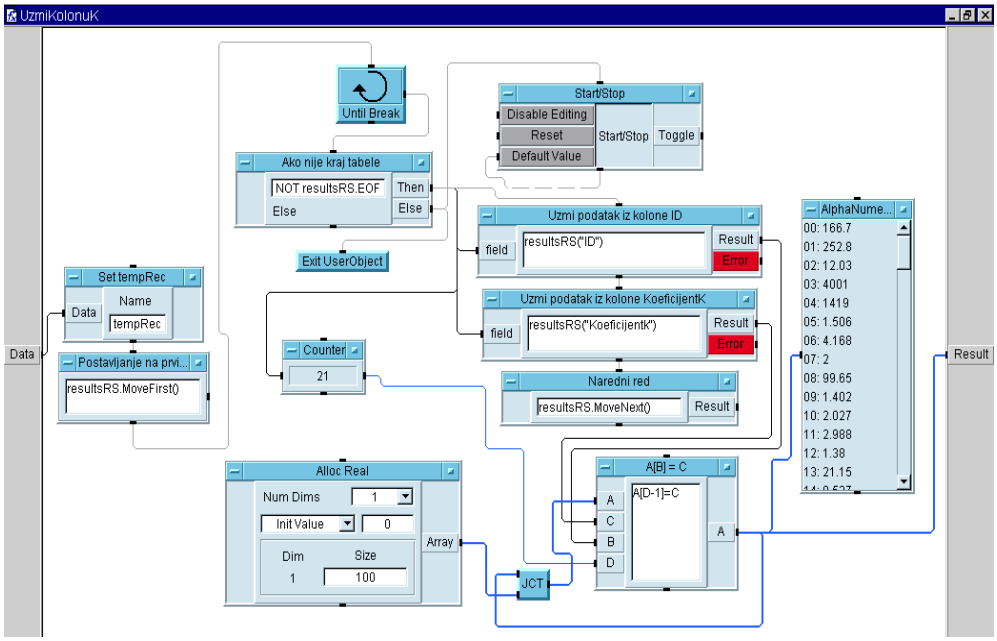
Slika 5.3 Deklarisanje promjenljivih za komunikaciju sa Accessom

Na slici 5.4 prikazan je dio programa, koji se koristi za upis i čitanje podataka iz već postojeće baze podataka "Data\\Baza97MM18.mdb" koji se nalazi na direktorijumu "c:\\Akvizicija\\MM18\\ProbaVee\\"+"\\Data\\Baza97MM18.mdb". Upis podataka u tabelu se vrši pojedinačno po kolonama, pri čemu treba navesti i kolonu tabele u koju se upisuje podatak (npr. resultsRS("T1")= 55, za upis broja 55 u kolonu pod nazivom T1, resultsRS("T2")= B, za upis vrijednosti sa ulaza B u kolonu pod nazivom T2). Upis se vrši obično dodavanjem novog reda na kraju tabele preko komande "resultsRS.AddNew()". Upis podataka u tabelu baze se može vršiti i preko objekta *Record array* pojedinačno po kolonama. Čitanje podataka iz baze je komplikovanije nego čitanje podataka iz *Excel* dokumenta.

Problem je u tome što podatak u bazi zavisi od kolone i reda zapisa, pri čemu red zapisa nije jednoznačno određeno kao u *Excelu*. Zato je poželjno da se preuzimanje podataka iz baze napravi u vidu jedne posebne funkcije kao na slici 5.5, u kojoj je prikazano preuzimanje svih podataka iz kolone "ID" i kolone "Koeficijentk" i sastavljanje u jedan dvodimenzioni niz. Na kraju rada sa bazom podataka, potrebno je prvo zatvoriti sve otvorene tabele sa komandom "*resultsRS.Close()*", a zatim i čitavu bazu podataka sa komandom "*mdb.Close()*".



Slika 5.4 Primjer programa za komunikaciju sa Accessom



Slika 5.5 Primjer funkcije za čitanje iz određene kolone

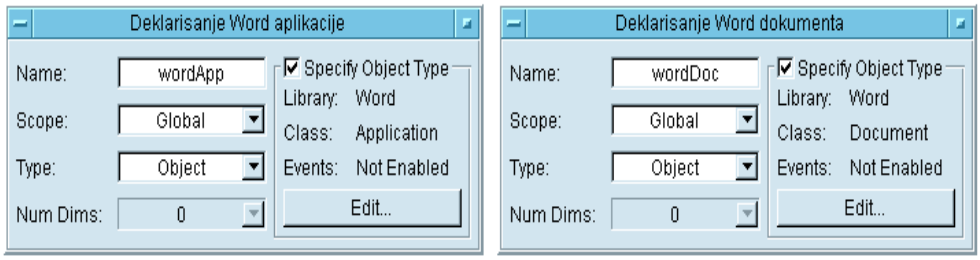
5.3 *MICROSOFT WORD*

Programski jezik *VEE Pro* ima mogućnost pravljenja programa, koji direktno komunicira sa dokumentom napravljenim u *Microsoft Wordu,* ali samo u verziji *Word97,* kako bi se ubrzala komunikacija između ova dva programa. Program napisan u *VEE Pro* najčešće komunicira sa tekstualnim dokumentom, tako što u njih upisuje rezultate mjerenja ili iz dokumenta preuzima neke predhodno snimljene podatke. Da bi se ova komunikacija ostvarila potrebno je prvo deklarisati promjenljive. Primjer deklarisanja promjenljivih prikazan je na slici 5.6. U ovom primjeru deklarisane su sljedeće promjenljive:

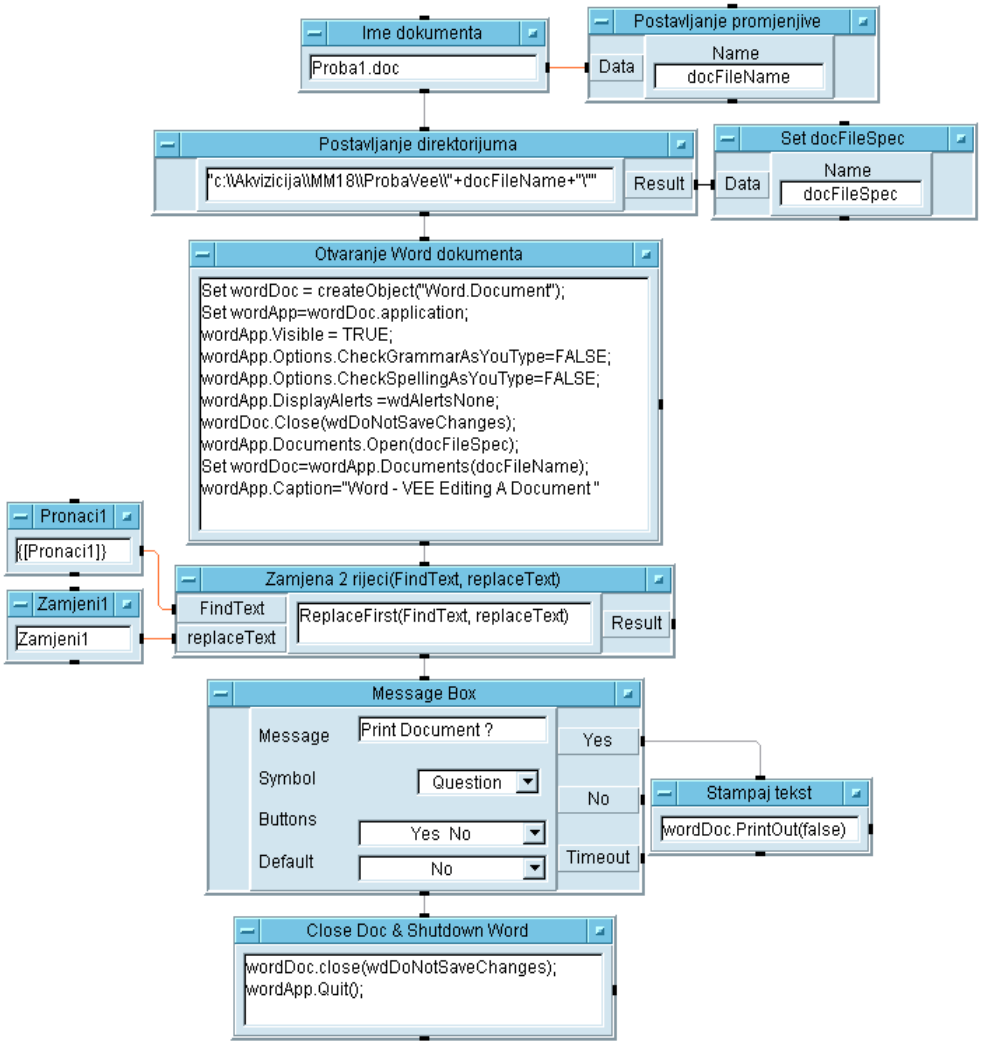
- *wordApp*
- *wordDoc*.

Promjenljiva ***wordApp*** namijenjena je za komunikaciju sa kompletnim *Word* programom. Mora biti deklarisana na globalnom nivou kako bi se mogla koristiti u glavnom programu, ali i svim podprogramima i funkcijama. Tip podatka treba deklarisati kao *Object*.

Promjenljiva ***wordDoc*** namijenjena je za komunikaciju sa jednim dokumentom. Ako koristimo više dokumenata tada u programu moramo imati posebno deklarisanu promjenljivu za svaki dokument. Ova promjenljiva mora biti deklarisana na globalnom nivou. Tip podatka treba deklarisati kao *Object*.



Slika 5.6 Deklarisanje promjenljivih za komunikaciju sa Wordom



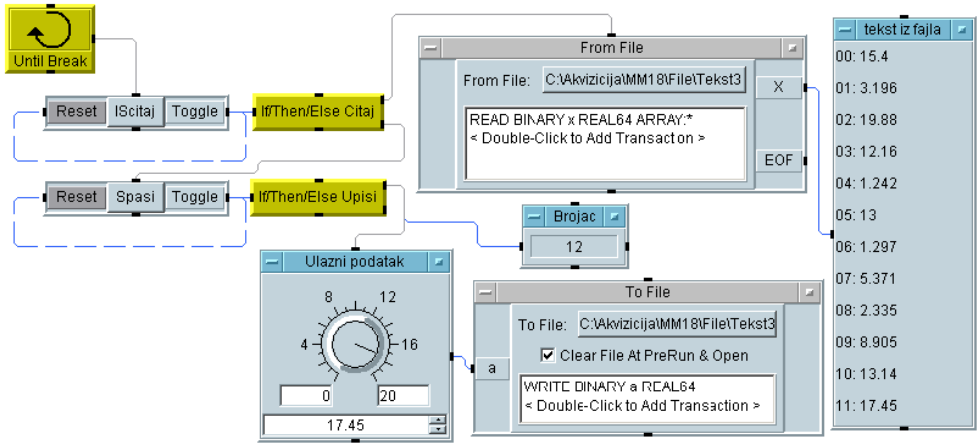
Slika 5.7 Primjer programa za komunikaciju sa Wordom

Na slici 5.7 prikazan je dio programa, koji se koristi za zamjenu nekog teksta u *Word* dokumentu, a zatim i pojavljivanje menija za štampanje tog dokumenta. Na kraju rada sa *Word* dokumentom potrebno je prvo zatvoriti sve otvorene dokumente sa komandom "*wordDoc.close(wdDoNotSaveChanges)*", a zatim i čitavu *Word* aplikaciju sa komandom "*wordApp.Quit()*".

5.4 POVEZIVANJE SA FAJLOM ZA ČUVANJE PODATAKA

Programski jezik *VEE Pro* ima mogućnost pravljenja programa, koji direktno komunicira sa dokumentima napravljenim u *Microsoft Wordu, Microsoft Excelu i Microsoft Accessu*. Svi ovi dokumenti imaju lijep grafički interfejs, međutim svi oni zazimaju više memorijskog prostora od običnog fajla za čuvanje podataka.

Kada se iz programa *VEE Pro* želi snimiti velika količina podataka na računaru, a da se pri tome ostvari i najveća moguća brzina prenosa podataka, najbolje je koristiti objekat *To File*. Ovaj objekat se koristi za snimanje svih tipova podataka u novi ili već postojeći fajl, koji se nalazi na tačno određenoj lokaciji na računaru. Ovi fajlovi se mogu direktno čitati pomoću programa *Notepad*. Objekat *From File* se koristi za čitanje podataka iz već postojećeg fajla, koji se nalazi na tačno određenoj lokaciji na računaru. Na slici 5.8 predstavljen je primjer u kome se prvo snimaju brojne vrijednosti u fajl *Tekst3*, a zatim se te snimljene vrijednosti čitaju i prikazauju na nekom indikatoru. Ako se nakon pokretanja programa podaci uvijek žele spašavati u potpuno prazan fajl, onda je potrebno u objektu *To File* čekirati opciju *Clear File At PreRun&Open*. Ako ova opcija nije čekirana, onda će prilikom svakog novog spašavanja podataka u fajlu ostati i sve stare snimljene vrijednosti. Pri tome se nove vrijednosti uvijek spašavaju iza starih.



Slika 5.8 Program za upis i čitanje iz fajla

5.5 *VISUAL BASIC*

Visula Basic je objektno orjentisan programski jezik. Posjeduje veliki broj gotovih objekata koji olakšavaju programiranje. Programski jezik *VEE Pro* se vrlo lako direktno povezuje sa programom napisanim u programskom jeziku *Visula Basic*. Pri tome se programski jezik *VEE Pro* najčešće koristi za pisanje uslužnih programa, koji se koriste za komunikaciju sa programabilnim mjernim instrumentima. Ti uslužni programi se uglavnom odnose na programiranje i saobraćaj mjernih podataka od računara prema mjernim instrumentima i od mjernih instrumenata prema računaru.

Sa druge strane u programskom jeziku *Visual Basic* obično je napravljen glavni program. Glavni program je deo korisničkog programa, koji opisuje operativni proces sistema i algoritam mjernog ciklusa. Glavni program određuje trenutak startovanja i zaustavljanja mjernog procesa, redoslijed mjernih ciklusa, broj mjerenja u jednom ciklusu i odredište mjernih podataka. U njemu se obavljaju matematičke operacije nad izmjerenim parametrima. Korisnik ima na raspolaganju jednostavan korisnički interfejs koji se pravi od velikog broja gotovih objekata koje *Visual Basic* ima. Na taj način je omogućeno jednostavno i brzo kretanje kroz program što je jako bitno prilikom postavljanja određenih preduslova za samo mjerenje.

Visual Basic takođe može direktno da komunicira sa svim programima iz *Microsoft Office* paketa, tako da se svi ovi programi i na taj načim mogu povezati sa programom napisanim u *VEE Pro*. Na ovaj način se značajno povećavaju mogućnosti programa *VEE Pro*.

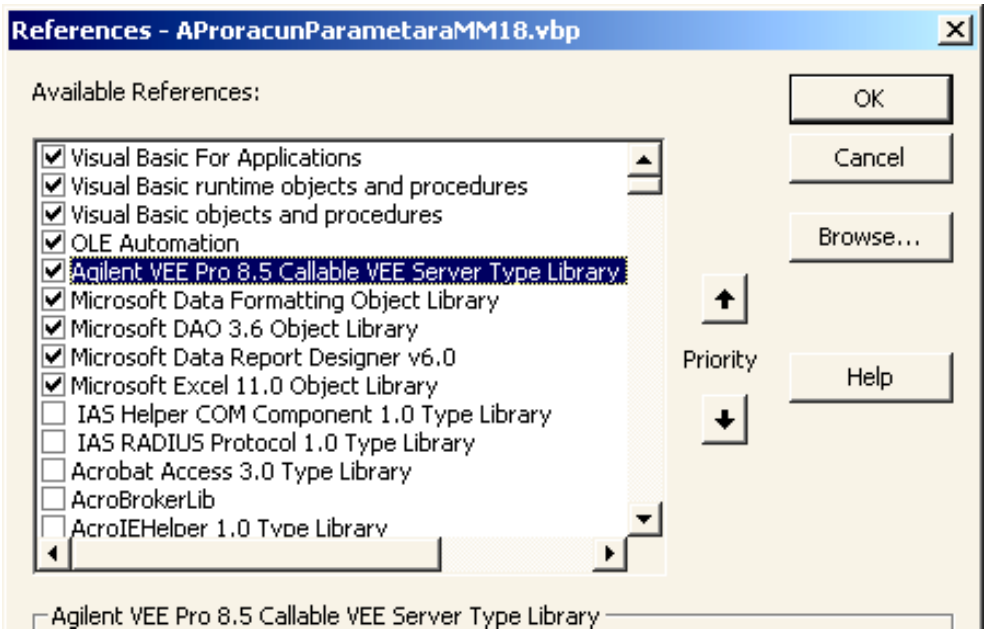
Da bi se *Visual Basicu* omogućilo povezivanje sa gotovim programa napisanim u programskom jeziku *VEE Pro*, potrebno je nakon pokretanja novog projekta u *Visual Basicu* (*File New → Project → StandardExe*), u glavnom meniju *Visual Basica* izabrati *Project → References*, nakon čega se pojavljuje prozor prikazan na slici 5.9. Na prozoru koji se pojavi potrebno je čekirati sve one programe sa kojima se želi direktno komunicirati iz programa koji se piše u *Visual Basicu.*

U primjeru programu koji je podešen preko prozora prikazanog na slici 5.9 može se direktno komunicirati sa:

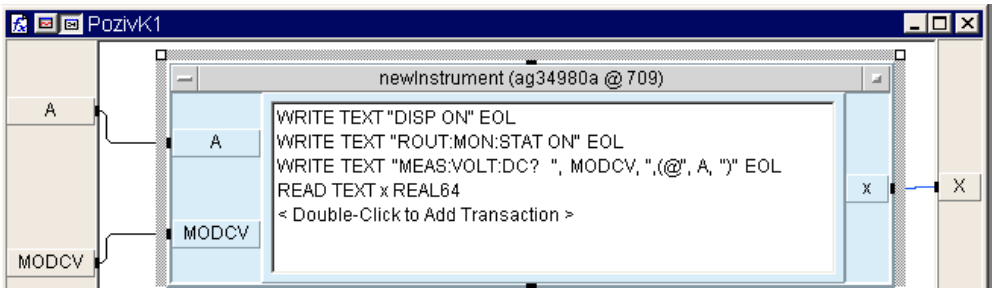
- programom napravljenim u programskom jeziku *VEE Pro* preko opcije *Agilent VEE Pro 8.5 Callable VEE Server Type Library,*

- bazom podataka napravljenom u *Microsoft Accesu* preko opcije *Microsoft DAO 3.6 Object Library,*

- dokumentom napravljenim u *Microsoft Excelu* preko opcije *Microsoft Excel 11.0 Object Library*.



Slika 5.9 Simulacija osciloskopa



Slika 5.10 Funkcija napisana u VEE Pro

Sada će biti predstavljen kod programa napisan u *Visual Basicu*, a koji omogućuje pozivanje i pokretanje programa "Etalon3.vee" napisanog u programskom jeziku *VEE Pro*. Program "Etalon3.vee" se koristi za daljinsko mjerenje napona pomoću mjerno akvizicionog sistema, nakon čega se izmjerena vrijednost šalje u promjenljivu programa napisanog u *Visual Basicu*. Ovaj program napisan u *VEE Pro* ima u sebi posebnu funkciju "PozivK1"koja se koristi za mjerenje. Ova funkcija ima dvije ulazne i jednu izlaznu promjenljivu, a njen izgled prikazan je na slici 5.10. U kodu ovog programa je predstavljeno i povezivanje *Visual Basica* sa bazom podataka napravljenom u *Accessu*. Komentari koji opisuju naredbe u kodnim linijama programa, biće navedeni italik slovima iza ključne riječi *Rem*, koja se u *Visual Basicu* koristi za odvajanje komentara od korisnog koda.

Private Sub Form_Load()

Rem Deklaracija procedure koja se pokreće čim se pokrene forma.

Dim Baza As Database

Rem Deklarisanje promjenljive **Baza**, *koja se koristi za povezivanje Visual Basica sa bazom podataka u Accessu.*

Dim Zapis As Recordset

Rem Deklarisanje promjenljive **Zapis**, *koja se koristi za povezivanje Visual Basica sa tabelom u bazi podataka u Accessu.*

Dim Veza As New CallableVEE.CallServer

Rem Deklarisanje promjenljive **Veza**, *koja se koristi za povezivanje sa programom VEE Pro.*

Dim Biblioteke As CallableVEE.Library

Rem Deklarisanje promjenljive **Biblioteke,** *koja se koristi za povezivanje sa programom napisanim u programskom jeziku VEE Pro.*

Dim FunkcijaSVE4 As CallableVEE.UserFunction

Rem Deklarisanje promjenljive **FunkcijaSVE4**, *koja se koristi za pozivanje sa funkcijom, koja se nalazi unutar glavnog programa napisanog u programskom jeziku VEE Pro.*

Set Biblioteke = Veza.Libraries.Load(App.Path & "\Etalon3.vee")

Rem Uspostavljanje veze između promjenljive **Biblioteke** *i programa pod imenom* **Etalon3.vee**, *koji je napisan u VEE Pro. Ovaj program se nalazi na istom direktorijumu, gdje i program napisan u Visual Basicu.*

Set FunkcijaSVE4 = Biblioteke.UserFunctions("PozivK1")

Rem Uspostavljanje veze između promjenljive **FunkcijaSVE4** *i funkcije pod imenom* **PozivK1**, *koja se nalazi u programu pod imenom* **Etalon3.v**ee *koji je napisan u programskom jeziku VEE Pro.*

Set Baza = DBEngine.Workspaces(0).OpenDatabase _
(App.Path & "\Baza97MM18D.mdb")

Rem Uspostavljanje veze između promjenljive **Baza** *i baze podataka napravljene u Accessu pod imenom* **Baza97MM18D.mdb**. *Ova baza se nalazi na istom direktorijumu, gdje se nalazi i program napisan u Visual Basicu.*

Set Zapis = baza.OpenRecordset("Etaloniranje")

Rem Uspostavljanje veze između promjenljive **Zapis** *i tabele* **Etaloniranje**, *koja se nalazi u bazi podataka pod imenom* **Baza97MM18D.mdb**.

End Sub

Rem Kraj procedure u kojoj je izvršena deklaracija promjenljivih.

Private Sub Tvred_Click()

Rem Deklaracija procedure, koja se pokreće klikom miša na dugme **Tvred**.

Dim tv, Broj, MOKanalaBP As Double

Rem Deklaracija promjenljivih, koje su realnog tipa podataka.

Dim j, kanalBP As Integer

Rem Deklaracija promjenljivih, koje su cjelobrojnog tipa podataka.

Dim MODCV As String

Rem Deklaracija promjenljivih, koje su tekstualnog tipa podataka.

Dim Izlaz(1) As Variant

Rem Deklaracija nizovne promjenljive **Izlaz(1)** *od dva člana, koji mogu biti bilo kog tipa podataka. Ova promjenljiva će se koristiti za slane podataka iz Visual Basica u program napisan u VEE Pro.*

Dim Ulaz(0) As Variant

Rem Deklaracija nizovne promjenljive **Ulaz(0)** *od jednog člana, koji može biti bilo kog tipa podataka. Ova promjenljiva će se koristiti za slanje podataka iz VEE Pro, u program koji se piše u Visual Basicu.*

kanalBP = Zapis("kanal")

Rem U promjenljivu **kanalBP** *se upisuje podatak iz kolone* **kanal***, tabele* **Etaloniranje***, baze podataka* **Baza97MM18D.mdb**.

Izlaz(0) = kanalBP

Rem U promjenljivu **Izlaz(0)** *se upisuje podatak iz promjenljive* **kanalBP***, koji se želi poslati u program VEE Pro.*

MOKanalaBP = zapis("MOKanala")

Rem U promjenljivu **MOKanalaBP** *se upisuje podatak iz kolone* MOKanala*, tabele* **Etaloniranje***, baze podataka* **Baza97MM18D.mdb**.

```
If MOKanalaBP = 0.1 Then
    MODCV = "0.1,0.0000001"
End If
If MOKanalaBP = 1 Then
    MODCV = "1,0.000001"
End If
If MOKanalaBP = 10 Then
    MODCV = "10,0.00001"
End If
If MOKanalaBP = 100 Then
    MODCV = "100,0.0001"
End If
```

Rem Četiri uslova za izbor odgovarajućeg mjernog opsega, koji se želi poslati u program VEE Pro.

Izlaz(1) = MODCV

Rem U promjenljivu **Izlaz(1)** *se upisuje podatak iz promjenljive* **MODCV***, koji se želi poslati u program VEE Pro.*

Veza.Left = 0

Rem Dodjeljivanje početne pozicije pojavljivanja programa VEE Pro sa lijeve strane ekrana u broju piksela.

Veza.Top = -20

Rem Dodjeljivanje početne pozicije pojavljivanja programa VEE Pro sa vrha u broju piksela, a -20 znači da neće biti vidljivo zaglavlje programa VEE Pro.

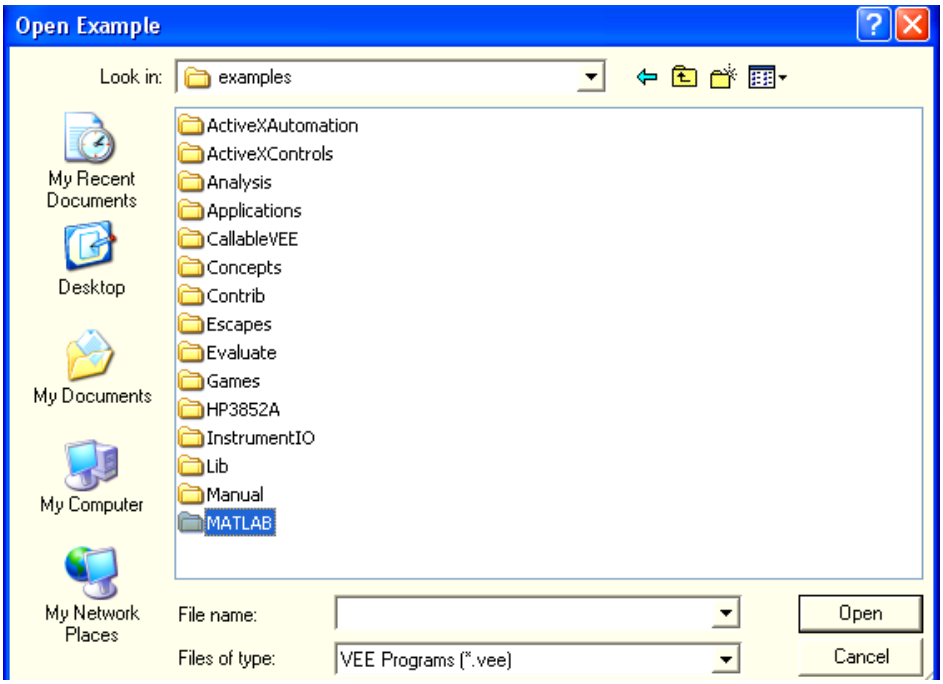
Veza.Height = 600
Rem Dodjeljivanje visine prozora programa VEE Pro u broju piksela.
Veza.Width = 800
Rem Dodjeljivanje širine prozora programa VEE Pro u broju piksela.
FunkcijaSVE4.Call Izlaz, Ulaz
Rem Poziv programa napisanog u VEE Pro pod imenom **FunkcijaSVE4**. *U ovaj program se šalju podaci preko promjenljive* **Izlaz**. *Kada program VEE Pro izvrši mjerenje on vraća rezultat preko promjenljive* **Ulaz**.
Broj = Ulaz(0)
Rem Promjenljiva **Broj** *preuzima vrijednost poslatu iz programa VEE Pro.*
TvNapona.Text = Broj
Rem Prikaz ove vrijednosti u tekst box.
k = Zapis("Koeficijentk")
Rem Upis podatka iz kolone **Koeficijentk** *u promjenljivu* m.
m = Zapis("Koeficijentn")
Rem Upis podatka iz kolone **Koeficijentn** *u promjenljivu* n.
tv = Broj * k + m
tv = Round(tv, 4)
trvred.Text = tv
Zapis.MoveFirst
Rem Kretanje kroz tabelu baze podataka koja je trenutno otvorena.
Do While zapis("ID") <> Izabranisignal
Zapis.MoveNext
Loop
Zapis.Edit
Rem Dozvoljavanje izmjene podataka u bazi podataka.
Zapis("PosEtaloniranje") = datum
Rem Upis podatka u kolonu **PosEtaloniranje** *trenutno aktivne baze podataka iz promjenljive* **datum**.
Zapis.Update
Rem Spašavanje izmjena koje su urađene u bazi podataka.
Zapis.Close
Rem Zatvaranje tabele postavljena pod imenom **Zapis** *u bazi podataka koja je trenutno aktivna.*
Set Zapis = Nothing
Rem Oslobađanje promjenljive pod imenom **Zapis**.
Baza.Close
Rem Zatvaranje kompletne baze podataka postavljene pod imenom **Baza**.
Biblioteke.Unload
Rem Raskidanje veze sa programom napisanim u VEE Pro.
End Sub
Rem Kraj procedure **Tvred_Click()**.

6 PRIMJERI *VEE PRO* PROGRAMA

U ovom poglavlju biće predstavljeni primjeri programa napisanih u programskom jeziku *VEE Pro*. Prvo će biti predstavljeni primjeri programa, koji se mogu naći u *Helpu* programskog jezika *VEE Pro*. Zatim će biti predstavljeni praktični primjeri kodova programa, koji služe za komunikaciju sa programabilnim mjernim instrumentima. Na kraju će biti predstavljeno par programa, koji se mogu koristiti za simulaciju na računaru, labaratorijskih vježbi iz predmeta Električna mjerenja.

6.1 PRIMJERI IZ HELPA *VEE PRO* PROGRAMA

U Helpu programskog jezika *VEE Pro* postoji dosta gotovih primjera programa, koji mogu poslužiti kao dobra osnova za učenje ovog programskog jezika. Svaki primjer ima već urađene gotove programske blokove, koji se mogu iskopirati u novi program, koji se pravi. Do ovih primjera se može doći preko explorera na direktorijumu **"C:\Program Files\Agilent\VEE Pro8.5\examples"**. Drugi način da se dođe do ovih primjera je preko padajućeg menija ***Help*** i opcije ***Open Example*.** Nakon izbora ove opcije pojaviće se prozor prikazan na sljedećoj slici 6.1.



Slika 6.1 Prozor sa direktorijumima gdje se nalaze gotovi primjeri programa

Na direktorijumu **ActiveX Automation** nalaze se gotovi primjeri programa, koji opisuju vezu programa *VEE Pro* sa drugim programima iz *Microsoft Office* okruženja. Predstavljeni su primjeri povezivanja sa *Access, Word* i *Excel* dokumentima. Takođe postoji mogućnost kreiranja potpuno novih dokumenata direktno iz *VEE Pro* programa. Posebnu pažnju treba obratiti na to da *VEE Pro* komunicira samo sa *Microsoft Office* dokumentima u verziji 97. Ovo predstavlja jedno veliko ograničenje prilikom korišćenja ovog programa. Ovo je urađeno kako bi se dobilo na brzini povezivanja sa ovim programima, jer se oni uglavnom koriste za smještanje rezultata mjerenja.

Na direktorijumu **ActiveX Controls** nalaze se gotovi primjeri, koji opisuju *ActiveX Controle,* koje omogućuju direktnu vezu programa *VEE Pro* sa drugim programima instaliranim na računaru.

- U primjeru **Calendar** prikazan je način, kako se može koristiti objekat kalendara za rad sa vremenom i datumima.
- U primjeru **CommonDialog** prikazan je način, kako se može napraviti standardni *Windows* dijalog prozor (*Open* ili *Save*).
- U primjeru **ListView** prikazan je način punjenja listi sa novim podacima.
- U primjeru **MapiMail** prikazan je način slanja e-mail poruka.
- U primjeru **PDFViwer** prikazan je način gledanja *PDF* dokumenata, koji postoje na nekom direktorijumu računara.
- U primjeru **ProcessBar** prikazan je način, kako može da se napravi i koristi objekat *Bar Graf*.
- U primjeru **RichTextBox** prikazan je način komunikacije sa tekst dokumentom, koji je napravljen u *Rich Text* formatu.
- U primjeru **WebBrowser** prikazan je način, kako se može pristupiti i komunicirati sa različitim internet adresama.
- U primjeru **WindowsMediaPlayer** prikazan je način, kako se mogu pokretati audio i video fajlovi preko programa *WindowsMediaPlayer*.

Na direktorijumu **Analysis** nalaze se gotovi primjeri, koji prikazuju simulaciju raznih analiza signala koje pruža program *VEE Pro*. Rezultati ovih analiza se prikazuju na raznim dijagramima.

- U primjeru **AnalogFilter** prikazan je način pravljenja analognog filtera za signal u obliku četvrtke.
- U primjeru **CurveFit** prikazan je način aproksimacije nekog mjernog signala sa eksponencijalnom, logoritamskom, linearnom funkcijom ili aproksimacija pomoću polinoma određenog stepena.
- U primjeru **FutValue** prikazan je način primjene finansijske funkcije *PV*, *PMT, RATE, NPER*, kao i drugih finansijskih funkcija, za predikciju ponašanja budućih novčanih tokova i njihovo grafičko predstavljanje.

- U primjeru **MagDist** prikazan je način korišćenja funkcije *MagDist()*, koja omogućuje pravljenje histograma i računanje raspodjele signala.
- U primjeru **Mixer** prikazan je način spajanja dva signala, različitih frekvencija iz virtualnih izvora (200 Hz i 5 kHz), u jedan signal. Spajanje se vrši množenjem dva signala, koji se dobijaju iz virtualnih izvora. Na ovaj način se dobija modulisani signal. Modulisani signal se zatim prikazuje u vidu *FFT* spektra.
- U primjeru **Psk** prikazan je način šifrovanja i dešifrovanja teksta.
- U primjeru **ReadMarkerValues** prikazano je predstavljanje vremenski promjenljivog signala na XY dijagramu. Zatim je opisan način, kako se na XY dijagramu mogu postaviti horizontalni i vertikalni markeri. Predstavljen je način očitanja digitalne vrijednosti koje marker ima, kao i prikaz razlike vrijednosti dva markera.
- U primjeru **Setmap** prikazan je način prikaza spektra slučajnog signala promjenljive frekvencije. Dat je primjer računanja amplitude i frekvencije na kojoj se pojavljuje maksimum signala u spektru.
- U primjeru **Traj1 i Traj2** prikazan je način računanja i grafičkog predstavljanja matematičkog izvoda i integrala funkcije (krive), koja obično predstavlja izmjerene vrijednosti mjernog signala.
- U primjeru **Xfftfunc** prikazano je grafičko predstavljanje *FFT* (brza furijerova transformacija) analize na spektralnom dijagramu, za više različitih signala.

Na direktorijumu **Applications** nalaze se gotovi primjeri, koji prikazuju kako se mogu riješiti neki praktičnini problemi, u različitim naučnim disciplinama.

- U primjeru **Beam** prikazan je program za razdvajanje svjetlosti na spektar boja i prikaz na XY dijagramu.
- U primjeru **Chaos** prikazan je program, koji obrađuje i prikazuje na XY dijagramu propulzivni model kretanja.
- U primjeru **Convert** prikazan je program za konverziju iz jednog u drugi brojni sistem (binarni, oktalni, heksadecimalni i dekadni).
- U primjeru **Digfilt** prikazan je program za pravljenje digitalnog filtera.
- U primjeru **Lissajou** prikazan je program za pravljenje lisažeove figure, koja se dobije dovođenjem signala iz dva virtualna generatora funkcije na objekat, koji predstavlja simulaciju osciloskopa.
- U primjeru **Pid** prikazan je način pravljenja *PID* algoritma prvog i drugog reda, za automatsku regulaciju u zatvorenoj petlji. Rezultat regulacije se automatski prikazuje na objektu, koji predstavlja simulaciju osciloskopa.
- U primjeru **Simp_npv** prikazan je način primjene finansijske funkcije za izračunavanje iznosa kredita, u funkciji broja otplatnih perioda.
- U primjeru **Telecomm** prikazan je način simulacije *DSI* testa u telekominikacijama.

- U primjeru **Torsion1 i Torsion2** prikazan je program za izračunavanje ugaonog pomjeraja kod kružnog kretanja. Primjer predstavlja rješavanje problema iz mehanike.
- U primjeru **Wizbang** prikazan je program u kome se kombinuje upotreba dugmadi, prekidača, listi, opcionih dugmadi i simulacije raznih analognih i digitalnih indikatora.

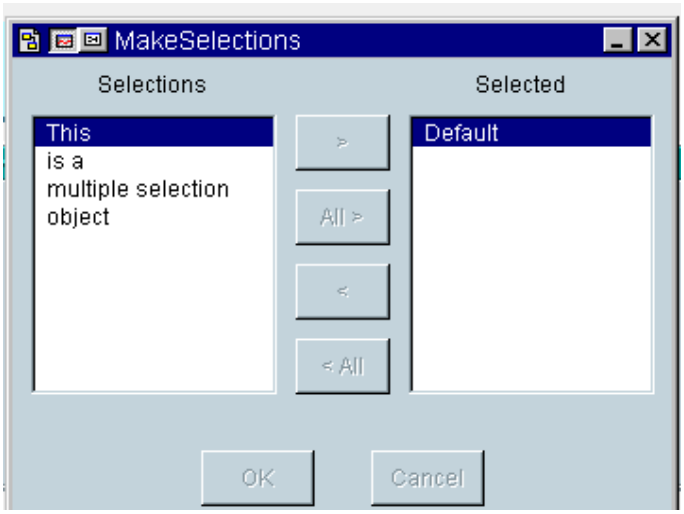
Na direktorijumu **Concepts** nalaze se gotovi primjeri, koji prikazuju koncepte u rješavanje nekih praktičnih problema.

- U primjeru **AdvanceDialog** prikazan je način pravljenja raznih dijalog prozora, pomoću kojih se od više ponuđenih, u jednom trenutku može izabrati samo jedna opcija.
- U primjeru **Ascii** prikazan je program, koji vrši konverziju slova i simbola u *ASCII* kod i obrnuto.
- U primjeru **Complex** prikazan je način predstavljanja kompleksnog broj na dijagramu u polarnom kordinatnom sistemu.
- U primjeru **Feedback** prikazan je način pravljenja povratne veze kod *IIR* digitalnog filtera.
- U primjeru **If** prikazan je program za prikaz "*IF THEN ELSE*" strukture uslovnog grananja.
- U primjeru **InfoAbout** prikazan je način pravljenja padajućeg menija, koji predstavlja pomoć pri korišćenju programa. Prikazano gdje i kako se unosi tekst, koji treba da nam pruži dodatne informacije vezane za program.
- U primjeru **Integrate** prikazan je program za računanje određenog integrala (između donje i gornje granice) zadate funkcije, a zatim i grafičko predstavljanje dobijene funkcije.
- U primjeru **MagPhase** prikazan je program za prikaz amplitude i faze naizmjeničnog signala u frekventnom spektru. Na dijagramu postoje i kursori za lakše i preciznije očitanje.
- U primjeru **Opt1** prikazan je program u kome se kombinuje upotreba podataka iz dva jednodimenziona niza, kao i način izdvajanja pojedinačnih elemenata niza.
- U primjeru **Opt2** prikazan je program za upotrebu podataka iz postojećeg dvodimenzionog niza, koji se nalazi u nekom fajlu. Opisana je primjena objekata za rad sa nizovima *Alloc Real Arrey* i *Set Values*.
- U primjeru **Redim** prikazan je program u kome se naknadno mijenja inicijalno postavljena dužina niza.
- U primjeru **SetXYMarker** prikazan je način pravljenja markera na XY dijagramu, pri čemu se marker automatski sam pomjera duž funkcije koja je prikazana na dijagramu.
- U primjeru **Strings** prikazan je program u kome su opisane funkcije za rad sa stringom (tekstom): *Len, Up, Dovn, Left* i *Right*.

- U primjeru **Text1** prikazan je program u kome su spajaju stringovi sa objektima *Concat* i *JCT* ili direktno pomoću operatora "+". Predstavljeno je i pravljenje jednodimenzionog niza (*Array*) od tekstualnih podataka.
- U primjeru **Timetick** prikazan je program za generisanje sistemskog vremena i datuma u *VEE Pro* programu. Opisano je kako se oni dalje mogu razlagati na sastavne dijelove i koristiti dalje u programu.
- U primjeru **Tofrom1** prikazan je program za pravljenje tekstualnog dokumenata (fajla) na nekom određenom direktorijumu i način upisa podataka u taj dokument.
- U primjeru **Tofrom4** prikazan je program u kome se signal (po amplitudi i vremenu) zapisuje u razne oblike tekstualnih dokumenata i prikazuje na alfanumjeričkom displeju (displej za prikaz brojne vrijednosti).
- U primjeru **Zerofill** prikazan je program u kome se povećava broj članova niza, a novonastalim članovima se dodjeljuje vrijednost nula.

Na direktorijumu **Contrib** nalaze se gotovi primjeri, koji prikazuju povezivanje sa *Excelom*, *SQL* bazom podataka i raznim interfejsima.

- U primjeru **ExcelAsDatabase** prikazan je način komuniciranja sa *Excel* dokumentom.
- U primjeru **InstrumentAssistant** prikazan je program, koji pronalazi sve instrumente spojene preko *GPIB* interfejsa.
- U primjeru **MaceSelections** prikazan je program, koji obezbjeđuje pravljenje dva prozora za prebacivanje podataka sa jednog mjesta na drugo i obrnuto kao na slici 6.2.
- U primjeru **SQLServerUsingADO** prikazan je način povezivanja sa bazom podataka napravljenom u verziji *SQL Server 7.0*.



Slika 6.2 Program za prebacivanje podataka

Na direktorijumu **Evaluate** nalaze se gotovi primjeri, koji prikazuju način korišćenja nekih gotovih objekata koje ima *VEE Pro*.

- U primjeru **Displeys** prikazanje način korišćenja raznih vrsta displeja za prikaz brojnih vrijednosti, koji postoje u programu *VEE Pro*.
- U primjeru **DLL** prikazan je program, koji pokazuje kako se poziva funkcija *DLL*.
- U primjeru **MaxVal** prikazan je program za pronalaženje maksimalog elementa niza.
- U primjeru **Select** prikazano je više načina selekcije, odnosno izbor samo jedne od više ponuđenih opcija.
- U primjeru **Togle** prikazan je program, koji koristi komandnu dugmad, razne prekidače, check boxs, signalne sijalice i kontinualne generatore.

Na direktorijumu **Games** nalaze se gotovi primjeri nekoliko igrica napravljenih u programu *VEE Pro*.

- U primjeru **Battleship** prikazana je igrica potapanja brodova.
- U primjeru **BlackJack, Euchre, Solitare** prikazane su igrice sa kartama.
- U primjeru **MineSveeper** prikazana je igrica pronalaženja mina postavljenih u minskom polju.
- U primjeru **Race** prikazana je igrica trke konja na hipodromu.
- U primjeru **Slots** prikazana je igrica sa slot aparatom na sreću.
- U primjeru **Wheel** prikazana je igrica točak sreće.

Na direktorijumu **HP3852A** nalaze se gotovi primjeri komunikacije sa nekim *Hewlettt Packard* mjernim instrumentima.

- U primjeru **GPIO1** prikazan je program za komunikaciju sa mjerno-akvizicionim sistemom HP3852A. Izmjerene vrijednosti se prikazuju na digitalnom displeju i smještaju u tekstualni fajl.
- U primjeru **GPIO2** prikazan je program za komunikaciju sa mjerno-akvizicionim sistemom HP3852A. Izmjerene vrijednosti se prikazuju na XY dijagramu i smještaju u tekstualni fajl.

Na direktorijumu **InstrumentIO** nalaze se gotovi primjeri komunikacije sa raznim programabilnim mjernim instrumentima. Programi iz ovog direktorijuma se ne mogu pokrenuti sve dok se računar ne spoji sa instrumentom, koji je predmet komunikacije.

- U primjeru **8720lims** prikazan je program za upisivanje nekih parametara u programabilni mjerni instrument.
- U primjeru **DIOFreq** prikazan je program za komunikaciju sa više programabilnih mjernih instrumenata.
- U primjeru **Gpibtest** prikazan je program za pronalaženje programabilnih uređaja priključenih preko *GPIB* interfejsa.

Na direktorijumu **Lib** nalaze se gotovi primjeri rada sa nizovima i dijagramima.

- U primjeru **1Dto2D** prikazan je program za prebacivanje podataka iz jednodimenzionog niza u dvodimenzioni niz (matricu - tabelu).
- U primjeru **2Dto1D** prikazan je program za prebacivanje podataka iz dvodimenzionog niza u jednodimenzioni niz.
- U primjeru **Barcht1** prikazan je program za prikazivanje podataka na *BarChat* dijagramu.
- U primjeru **Keypad** prikazan je program za unosa podataka preko simulacije tastature na ekranu.
- U primjeru **Piechart** prikazan je program za prikazivanje podataka na *PieChat* dijagramu.
- U primjeru **XYcntrl** prikazan je program za prikazivanje više različitih signala na XY dijagramu, koji ima više Y osa.

Na direktorijumu **Manual** nalaze se gotovi primjeri za praktično rješavanje upisa, čitanja i obrade izmjerenih podataka.

- U primjeru **Animate** prikazan je program, koji omogućuje kretanja slike i objekata po ekranu.
- U primjeru **Color** prikazan je program za podešavanje boje teksta i pozadine displeja na kojima se prikazuju rezultati.
- U primjeru **Dialog1** prikazan je program za unos podataka pomoću objekta *Text box*.
- U primjeru **Manual01** prikazan je program za korišćenja *IF THEN ELSE* strukture grananja.
- U primjeru **Manual02** prikazan je program, koji pomoću petlje *On Cycle* omogućuje neprekidno očitanje izmjerenih vrijednosti sa programibilnog instrumenta svake sekunde.
- U primjeru **Manual04** prikazan je program za korišćenje objekta *Sample&Hold* (čekanje) prije sabiranja u objektu formula. Ovo je neophodno, kada se čeka završenje neke petlje (*FOR* ili *While*) da bi se izvršila neka formula sa dva ili više ulaza.
- U primjeru **Manual06** prikazan je program, koji omogućuje punjenje jednodimenzionog niza A[i] preko povratne sprege, u kojoj se taj isti niz nalazi.
- U primjeru **Manual07** prikazan je program, koji omogućuje punjenje dvoodimenzionog niza A[i,j] preko povratne sprege, u kojoj se taj isti niz nalazi.
- U primjeru **Manual08** prikazan je program, koji koristi objekat *Get Values* za preuzimanje pojedinačnih podataka iz niza.
- U primjeru **Manual11** prikazan je program, koji koristi objekat *Get Field* (*rec.field*), kako bi se iz zapisa (objekat *Record* - tabela) razdvojili podaci, koji su različitog tipa (npr. broj od datuma).

- U primjeru **Manual13** prikazan je program, koji koristi objekat *Set Field* (*rec.field*=A) kako bi se u zapisu (*Record* - tabela) promijenilo ime jednog polja zapisa, a zatim i upisala nova vrijednost zapisa.
- U primjeru **Manual14** prikazan je program, koji koristi objekat *Get Field* kako bi se iz zapisa izdvojio neki element zapisa.
- U primjeru **Manual18** prikazan je program za deklarisanje globalne promjenljive, koja se poslije koristi u više naknadno napravljenih funkcija.
- U primjeru **Manual20** prikazan je program za spašavanje podataka u dokument, koji već postoji na nekom direktoriju. Poslije se preuzimaju podaci iz tog dokumenta i prikazuju na XY dijagramu.
- U primjeru **Manual21** prikazan je program za spašavanje podataka u tri kolone, u dokument koji već postoji na nekom direktoriju.
- U primjeru **Manual22** prikazan je program za spašavanje podataka, koji su rezultat mjerenja sa nekim instrumentom, u dvije kolone, u dokument koji već postoji na nekom direktoriju. Koristi se objekat *Collector*.
- U primjeru **Manual26** prikazan je program u kome se koristi objekat *Exit UserObject* za izlazak iz nekog podprograma ili funkcije ranije.
- U primjeru **Manual27, Manual29, Manual30** prikazan je program za čitanja nekih podataka iz fajla, koji postoji na nekom direktorijumu.
- U primjeru **Manual37** prikazan je program u kome se koristi objekat *Reise Error*, kako bi se prikazale eventualne greške, koje mogu nastati u toku izvršenja programa.
- U primjeru **Manual38** prikazan je program u kome se koristi objekat *UnBuild Record* za preuzimanje podataka tipa record (tabele sa poznatim nazivima kolona).
- U primjeru **Manual40, Manual41** prikazan je program u kome se koriste objekti *UnBuild Data, Data Set* i *From Data Set* za upis, čitanje podataka u neki fajl i prikaz na dijagramima.
- U primjeru **Manual43** prikazan je primjer pozivanja novo napravljene funkcije sa *Coll Function* iz glavnog programa.
- U primjeru **Manual44** prikazan je program u kome se koristi objekat *Colector* za spajanja dvije vrijednosti u jednu.
- U primjeru **Status** prikazan je program za korišćenje naredbe *ShowPanel* za prikaz nekog objekta ili mjernih rezultata na posebnom ekranu.
- U primjeru **Timeout** prikazan je program za prikaz različitih vrijednosti preko objekta *Dialog Box*, kada istekne neko kontrolno vrijeme.

Na direktorijumu **Manual/UserGuide** nalaze se veliki broj gotovih primjera za praktično rješavanje problema pomoću programa *VEE Pro*.

- U primjeru **AppA_applebagger2** prikazan je program za korišćenje objekta *Rondum* (generisanje slučajnog broja u rasponu od 0 do 1), *Accumulator* (sabiranje svih ulaznih vrijednosti), *Counter* (brojanje ulaznih

vrijednosti), *Shift registar* (pamćenje zadnjih nekoliko ulaza) i *Gate* (zaustavljanje ili propuštanje neke vrijednosti).

- U primjeru **AppA_Mask_test1** prikazan je program za poređenje mjernih signala sa predhodno zadatim brojnim vrijednostima. Ulazni signal i rezultat poređenja se prikazuju na istom XY dijagramu.
- U primjeru **AppA_Movedata_tofrom** prikazan je program za računanje srednje vrijednosti i standardne devijacije signala dobijenog iz generatora slučajnih brojeva. Signal i izračunate vrijednosti se zapisuju u jedan tekstualni fajl, a zatim se spašeni podaci pojedinačno čitaju iz tog fajla.
- U primjeru **AppA_panels_popups1** prikazan je program za unos podataka preko *Text boxa,* koji se pojavljuje sam na posebnom ekranu za unos podataka (slično kao objekat *Input box* iz Visual Basica).
- U primjeru **AppA_random_generator1** prikazan je program za korišćenje generatora slučajnih brojeva, pri čemu se računa minimalana i maksimalna vrijednost niza slučajnih brojeva.
- U primjeru **AppA_records1** prikazan je program za pravljenje višestrukog niza, a zatim spajanje dva niza u jedan.
- U primjeru **AppA_strings_globals** prikazan je program za rad sa stringom, u kome se izdvaja samo prezime iz teksta, koji čini ime i prezime.
- U primjeru **AppA_testing_numbers2 i 3** prikazan je program za pravljene raznih tipova tekstualnih poruka.
- U primjeru **AppA_userfunctions1** prikazan je program za pozivanje posebno napisane funkcije sa naredbom *Coll* iz glavnog (*Main*) programa.
- U primjeru **AppA_stats1** prikazan je program za pozivanja iz glavnog (*Main*) programa posebno napisane funkcije, koja ima 1 ulaznu promjenljivu i 4 izlazne promjenljive.
- U primjeru **Ch2_formula_objectprogram** prikazan je program za izdvajanja samo pozitivnih perioda iz sinusnog signala i prikaz na XY dijagramu. Amplituda signala se pri tome pomjera za neki *offset* po Y opsi.
- U primjeru **Ch2_using_datatypes** prikazan je program za sabiranje realnog (ili cjelobrojnog) broja i kompleksnog broja.
- U primjeru **Ch6_excel_example** prikazan je program za otvaranje postojećeg *Excel* dokumenta, koji zatim može da se spasi pod drugim imenom na proizvoljni direktorijum.
- U primjeru **Intro_MATLAB_Script_in_VEE** prikazan je program za prikazivanje kompleksnog broja (koji ima realni i imaginarni dio) na trodimenzionalnom dijagramu (realna osa, imaginarna osa i osa za prikaz amplitude) preko *MATLAB* skripta.
- U primjeru **Progress_bar** prikazan je program za pravljenje bargrafa.
- U primjeru **Report** prikazan je program za štampanja izvještaja, koji se štampa iz dokumenta, koji je spašen negdje na disku.

- U primjeru **Seqdat4** prikazan je program za korišćenje objekta *Sequencender*, koji omogućuje da se preko njega direktno poziva neka posebno napisana funkcija i da se izračunate vrijednosti dalje proslijeđuju.
- U primjeru **Uflab** prikazan je program za spajanje sinusnog signala dobijenog iz generatora funkcija i signala iz generatora slučajnih brojeva. Dobija se sinusni signal na koji je dodat šum po amplitudi.

Na direktorijumu **MATLAB** nalaze se gotovi primjeri, koji omogućuju korišćenje mogućnosti *Matlab* programa iz *VEE Pro* programa.

- U primjeru **BasicGraphManipulation** prikazan je program za grafičko predstavljanja signala u *MatLabu*.
- U primjeru **Interpolation** prikazan je program za interpolaciju (spajanje) dvije ili više tačaka, uz odabir načina spajanja, pomoću *MatLaba*.
- U primjeru **Logo3Dplot** prikazan je program za trodimenzioni prikaza slike nekog dijagrama.
- U primjeru **MultiGraphFigure** prikazan je program za prikazivanje dva različita XY dijagrama, jedan ispod drugog.
- U primjeru **MultiTraceGraph** prikazan je program za prikazivanje više različita signala (sa više Y osa) na jednom dijagramu.
- U primjeru **Surf3Dplot** prikazan je program za trodimenzioni prikaz slike, pri čemu se slika mijenja u realnom vremenu.

Na direktorijumu **DotNET** nalaze se gotovi primjeri programa, koji omogućuju nova podešavanja i gledanje ranije podešenih parametara računara.

- U primjeru **DataBaseReadUsingDataSet** prikazan je program, koji radi *SELECT* upite nad bazom i pravi rezervne kopije tabela.
- U primjeru **DeleteDir** prikazan je program za pravljenje novih i brisanja postojećih direktorijuma na diskovima računara.
- U primjeru **DiskInfo** prikazan je program za prikaz osnovnih podataka o hard diskovima računara na kome je program *VEE Pro* instalisan.
- U primjeru **EmerateEnums** prikazan je program, koji daje osnovne podatke o fajlovima na računaru.
- U primjeru **FileInfo** prikazan je program, koji otvara prozor za izbor fajlova, a zatim da za izabrani fajl daje osnovne podatke o njemu.
- U primjeru **Hastable** prikazan je program, koji radi spajanje i pretraživanje nizova pomoću funkcije *Hastable*.
- U primjeru **IPAddress** prikazan je program za pronalaženje IP adrese računara.
- U primjeru **PassWordMenagment** prikazan je program za pronalaženje ranije definisanih pristupa računaru i promjena šifre na pristupima.
- U primjeru **SelectFiles** prikazan je program za pravljenje prozora za pretragu i izbor fajlova.

- U primjeru **SelectFolder** prikazan je program za pregled i pronalaženje direktorijuma na računaru.
- U primjeru **SendMail** prikazan je program za pravljenje meil poruka.
- U primjeru **TimeZone** prikazan je program za pronalaženje vremenske zone, koja je podešena na računaru i osnovnih podataka koji prate to podešavanje.

Na direktorijumu **ProgrammaticProperty** nalaze se gotovi primjeri programa, koji prikazuju mogućnosti automatskog podešavanja nekih osobina objekata.

- U primjeru **ProgrammaticPropertyDemo** prikazan je program, koji omogućuje automatsku promjenu boje teksta, boje pozadine, veličine panela, mjesta pojavljivanja panela i objekata.
- U primjeru **SelectControlPropertyDemo** prikazan je program, koji omogućuje automatsku promjenu osobina objekta za višestruku selekciju.
- U primjeru **TabIndexChangeDemo** prikazan je program, koji omogućuje kretanje kroz text boksove za unos i prikaz podataka preko *Tab* tipke na tastaturi.

6.2 KOMUNIKACIJA SA PROGRAMABILNIM UREĐAJIMA

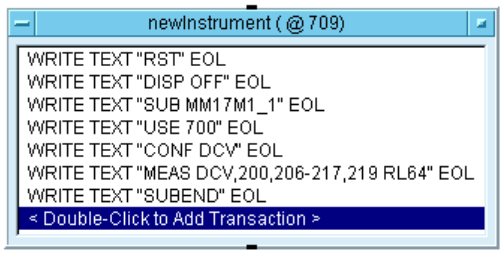
U ovom poglavlju biće predstavljeni neki primjeri koda programa, koji se koriste za komunikaciju računara sa nekim programabilnim mjernim uređajima. Svaki tip uređaja uglavnom ima različite komande, što malo otežava programiranje. Sve naredbe koje se koriste za programsko upravljanje sa uređajima, nalaze se u tehničkoj dokumentaciji uz te uređaje. Kod programa za komunikaciju sa uređajima upisuje se u objekat, koji se pronalazi preko menija *Instrument Menager*. Preko ovog menija se prvo pronalazi interfejs preko koga je programabilni uređaj trenutno povezan sa računarom. Kada se izabere interfejs pronalaze se svi programabilni uređaji, koji su u tom trenutku uključeni i preko interfejsa povezani sa računarom. Programabilni instrumenti i uređaji se mogu spajati sa računarom preko sljedećih interfejsa:

- ***GPIB***
- ***Serial***
- ***GPIO***
- ***USB/GPIB***
- ***Lan*** (samo za verziju programa *VEE Pro* 8.5 i novije verzije).

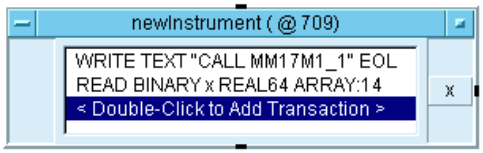
Pošto se na jedan računar može istovremeno spojiti više programabilnih uređaja, onda svaki uređaj ima svoju jedinstvenu adresu preko koje se vrši povezivanje. Dodavanje objekta za komunikaciju sa tim uređajem u program vrši se za verziju 6.0 klikom na dugme *Direct I/O*, a za verziju 8.5 i novije duplim

klikom na aktivni instrument. Kada se na radnu površinu za pisanje programa postavi novi objekat za komunikaciju sa programabilnim uređajem u njemu nema nikakvog koda. Tada se na osnovu dokumentacije proizvođača piše posebno kod za svaki vid komunikacije sa uređajem. Sve komande koje idu od računara prema uređaju (npr. izbor mjerne veličine i mjernog opsega) počinju sa ključnom riječi "*WRITE TEXT*". Zatim se željena komanda upisuje između navodnika, a na kraju reda svake komande stoji ključna riječ "*EOF*". Sve komande koje idu od uređaja prema računaru (npr. očitanje izmjerenih vrijednosti) počinju sa ključnom riječi "*READ*". Nakon nje se upisuje format podataka, koji se šalju iz uređaja (*BINAY* ili *TEXT*). Zatim je potrebno da se navede količina podataka koja se šalje, a na kraju reda svake komande stoji ključna riječ "*EOF*". Podaci se iz uređaja mogu slati jedan po jedan ili istovremeno više podataka u obliku niza (*Arrey*).

U jednom programu može postojati više objekata za komunikaciju sa istim uređajem, ali i više objekata za komuniokaciju sa različitim uređajima, koji su u istom trenutku povezani sa računarom. Pri tome treba voditi računa o redoslijedu izvršenja objekata u programu, kao i redoslijedu komandi u pojedinim objektima. Ne smije se desiti da dođe do istovremenog slanja komandi koje su međusobno u suprotnosti i koje nije moguće istovremeno izvršiti. Na primjer, svaki programabilni multimetar u jednom trenutku može da mjeri samo jednu veličinu, ali u malim vremenskim razmacima (koji mogu da iznose reda mili sekunde) može da mjeri različite veličine (napon, struju, otpor itd.) na raznim mjernim opsezima. Pojedini programabilni uređaji koriste se za generisanje signala (razni kalibratori). Povezivanjem programabilnih kalibratora i programabilnih mjernih instrumenata omogućeno je automatsko etaloniranje u metrološkim laboratorijama. Na ovaj način se značajno smanjuju troškovi etaloniranja, a povećava tačnost zbog povećanja broja mjernih tačaka i eliminacije grubih grešaka pri zapisivanju i obradi rezultata.



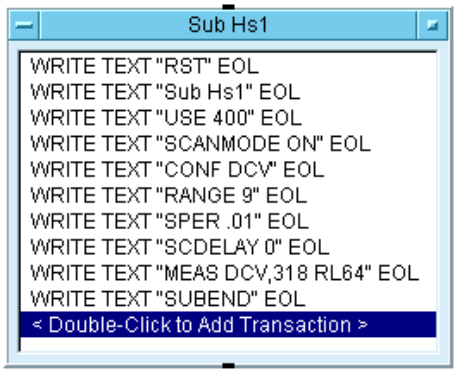
Slika 6.3 Podprogram za HP3852A



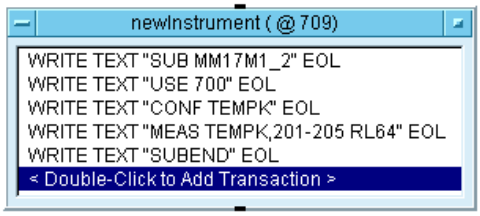
Slika 6.4 Poziv podprograma MM17M1_1

Na slici 6.3 prikazan je kod programa za mjerenje 14 različitih naponskih signala pomoću mjerno-akvizicionog sistema HP3852A, za vrijeme kraće od jedne sekunde. Ovaj kod je napisan kao podprogram, koji se smješta u internu memoriju mjerno-akvizicionog sistema. A ovaj podprogram se poziva pomoću drugog programa prikazanog na slici 6.4. Ovaj program se koristi za čitanje vrijednosti

izmjerenih pomoću podprograma. Mjerno-akvizicioni sistem HP3852A ima u sebi ugrađena dva različita multimetra. Jedan multimetar visoke klase tačnosti pored napona, struje i otpora može da mjeri i temperaturu pomoću otpornih termometara i termoparova, kao davača temperature. Prilikom mjerenja temperature pomoću termoparova vrši se automatska kompenzacija temperature hladnog spoja. Primjer pod programa za mjerenje temperature pomoću četiri termopara K tip prikazan je na slici 6.6. Drugi brži, ali multimetar manje klase tačnosti, koristi se za mjerenje brzo promjenljivih veličina ili za mjerenje više signala u kratkom vremenskom periodu. Primjer pod programa za mjerenje napona svakih 10 mili sekundi prikazan je na slici 6.5.

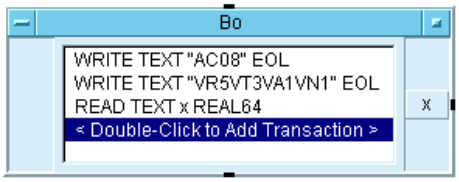


Slika 6.5 Mjerenje napona sa HP3852A

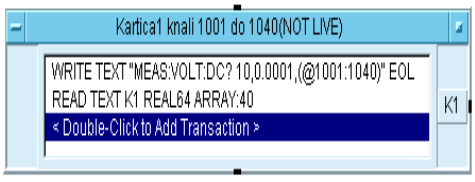


Slika 6.6 Mjerenje temperature sa HP3852A

Primjer programa za mjerenje napona pomoću mjerno-akvizicionog sistema HP3497A na jednom kanalu prikazan je na slici 6.7. Odmah nakon mjerenja izmjerena vrijednost napona se šalje na računar pomoću komande "*READ TEXT*" u izlaznu promjenljivu "X". Vrijednost ove promjenljive se zatim koristi u glavnom programu napisanom u *VEE Pro*.



Slika 6.7 Program za HP3497A

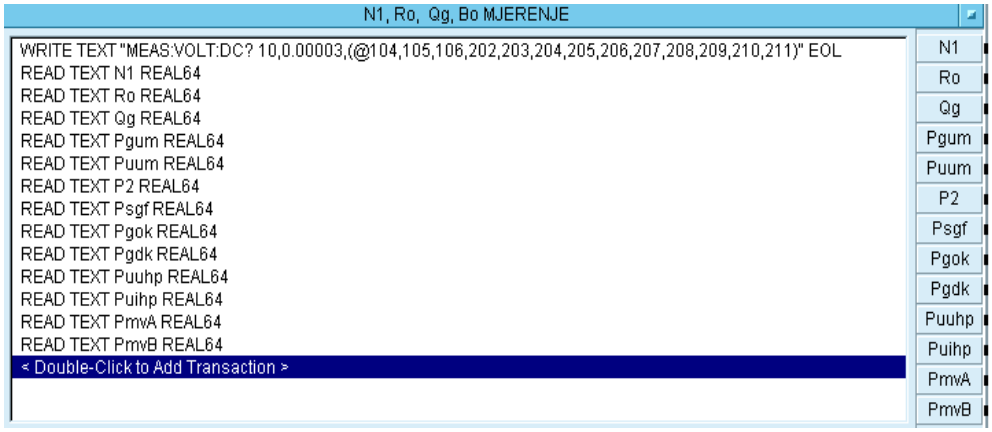


Slika 6.8 Program za Agilent 34980A

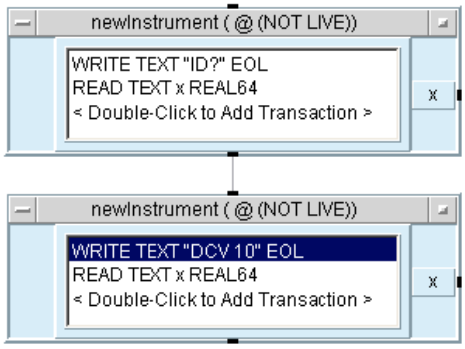
Primjer programa za mjerenje napona na četrdeset kanala jedne kartice pomoću mjerno-akvizicionog sistema Agilent 34980A, prikazan je na slici 6.8. Za vrijeme od pola sekunde se izmjeri napon, jedan za drugim, na svih četrdeset kanala. Kada se završi mjerenje na svim kanalima, sve izmjerene vrijednosti se šalju prema računaru u izlaznu promjenljivu "K1", u vidu niza od 40 realnih brojeva. U

programu *VEE Pro* se zatim iz ovog niza uzima jednu po jednu vrijednost svakog signala. Mjerno-akvizicioni sistem Agilent 34980A može da ima do osam ulaznih kartica. Postoje kartice sa po 70 kanala za mjerenje napona, struje, otpora ili temperature. Neke kartice su specijalizovane za mjerenje frekvencije. Pored kartica koje se koriste za mjerenje, na ovaj mjerno-akvizicioni sistem mogu se spojiti i kartice koje služe za generisanje napona, struje ili frekvencije.

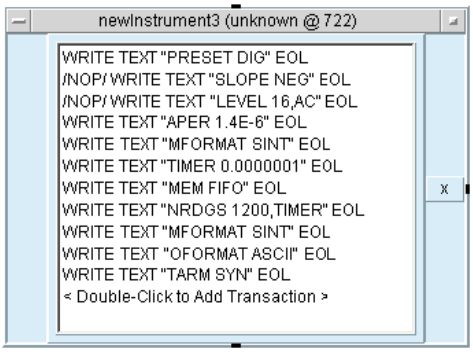
Primjer programa za mjerenje napona pomoću mjerno-akvizicionog sistema *Agilent* 34970A na trinaest kanala, koji se nalaze spojeni na dvije kartice, prikazan je na slici 6.9. Za vrijeme kraće od jedne sekunde izmjeri se napon, jedan za drugim, na svim kanalima. Kada se završi mjerenje svih kanala, sve izmjerene vrijednosti se šalju na računar, jedna po jedna u posebne izlazne promjenljive. Ovo je u odnosu na slanje podataka preko niza (vidi predhodni primjer), pregledniji način čitanja izmjerenih vrijednosti. Pogotovo je olakšano dalje pisanje programa za obradu mjernih rezultata, jer svakoj izlaznoj promjenljivoj možemo da damo ime prema nazivu signala koji mjerimo.



Slika 6.9 Program za Agilent 34970A



Slika 6.10 Program za HP3458A



Slika 6.11 Program za HP3458A

Snaga programskog jezika *VEE Pro* najviše dolazi do izražaja prilikom komuniciranja sa mjerno-akvizicionim sistemima, pomoću kojih se može mjeriti, ali i vršiti upravljanje nekim drugim uređajima. Međutim, ovaj program se koristi i za upravljanje mjernim instrumentima, koji imaju samo jedan ulaz za mjerne signale (na primjer digitalni multimetri). U primjeru na slici 6.10 prikazan je dio programa za komunikaciju sa digitalnim multimetrom HP3458A. Ovaj multimetar ima maksimalnu rezuluciju od 8 digita, a može da mjeri napon, struju, otpor i frekvenciju. Ovo je vrlo jednostavan program u kome se prvo traži identifikacija mjernog instrumenta pomoću komande "ID?", a zatim se uradi jednostavno mjerenje *DC* napona, sa automatskim izborom mjernog opsega. Kod programa prikazanog na slici 6.11 koristi se brzo mjerenje *AC* signala u obliku četvrtke frekvencije 100 kHz pomoću multimetra HP3458A. Izmjerene vrijednosti se smještaju u internu memoriju multimetra dok traje mjerenje, kako bi se povećala brzina mjerenja. Tek nakon završenog mjerenja izmjerene vrijednosti se šalju u glavni program. Ovo mjerenje je praktično nemoguće uraditi pomoću multimetra bez pomoći računara i programa *VEE Pro*.

6.3 MJERENJE PARAMETARA TURBOMLAZNOG MOTORA POMOĆU MJERNOAKVIZICIONOG SISTEMA

Jedan od najkompleksnijih problem mjerenja u vazduhoplovstvu je ispitivanje turbo-mlaznih motora (u daljem tekstu TMM-a) na stacionarnoj ispitnoj stanici. Stacionarne ispitne stanice su složeni namjenski objekti, na kojima se vrši simulacija svih radnih režima u kojima se motor može naći u toku leta aviona. Ispitivanje na stacionarnoj ispitnoj stanici predstavlja završnu aktivnost u procesu remonta TMM-a, pre njegove ugradnje na avion. Prilikom ispitivanja TMM-a na ispitnoj stanici mjeri se veliki broj različitih parametara. Ovi parametri se mjere u kratkom vremenskom periodu na motoru, postolju na koje je motor postavljen i ispitnoj ćeliji u kojoj se nalazi motor sa postoljem. Zato najbolje rješenje za zapisivanje (registrovanje) fizičkih procesa na ispitnoj stanici mora biti zasnovano na brzom računarski vođenom akvizicionom sistemu. Ovi mjerni sistemi odlikuju se kompleksnom strukturom i visokim mjernim performansama, koje moraju biti usklađene sa strogim kriterijuma propisanih kod testiranja vazduhoplovne opreme i motora predviđenih za ugradnju u savremene letjelice. U odnosu na broj mjernih signala koji se mjere na TMM-u prilikom leta aviona, na ispitnoj stanici se mjeri puno više mjernih signala na motoru i postolju na koje je motor postavljen. Tako se na ispitnoj stanici za ispitivanje TMM-a koji se ugrađuju u borbene avione Mig 21 nalaze 63 mjerna instrumenta, a na ispitnoj stanici za ispitivanje TMM-a koji se ugrađuju u borbene avione Galeb i Orao nalaze 34 mjerna instrumenta. Prilikom ispitivanja TMM-a na ispitnoj stanici ispitivač stalno prati i zapisuje pokazivanje

20 do 30 mjernih instrumenata, zavisno od režima rada motora. Pokazivanje nekih mjernih instrumenata ispitivač prati samo povremeno, dok pokazivanje nekih mjernih instrumenata ispitivač prati samo prilikom postavljanja TMM-a na ispitno postolje i prilikom konzervacije TMM-a, a poslije završenog ispitivanja. Neki režimi rada TMM-a traju kratko. Ispitivač nije u mogućnosti da u istom trenutku prati i ručno zapiše sve parametre potrebne za donošenje odluke u skladu sa kriterijumima prihvatljivosti datim u specifikaciji proizvođača TMM-a. Zato se javlja veliki subjektivni uticaj ispitivača na rezultate mjerenja parametara motora.

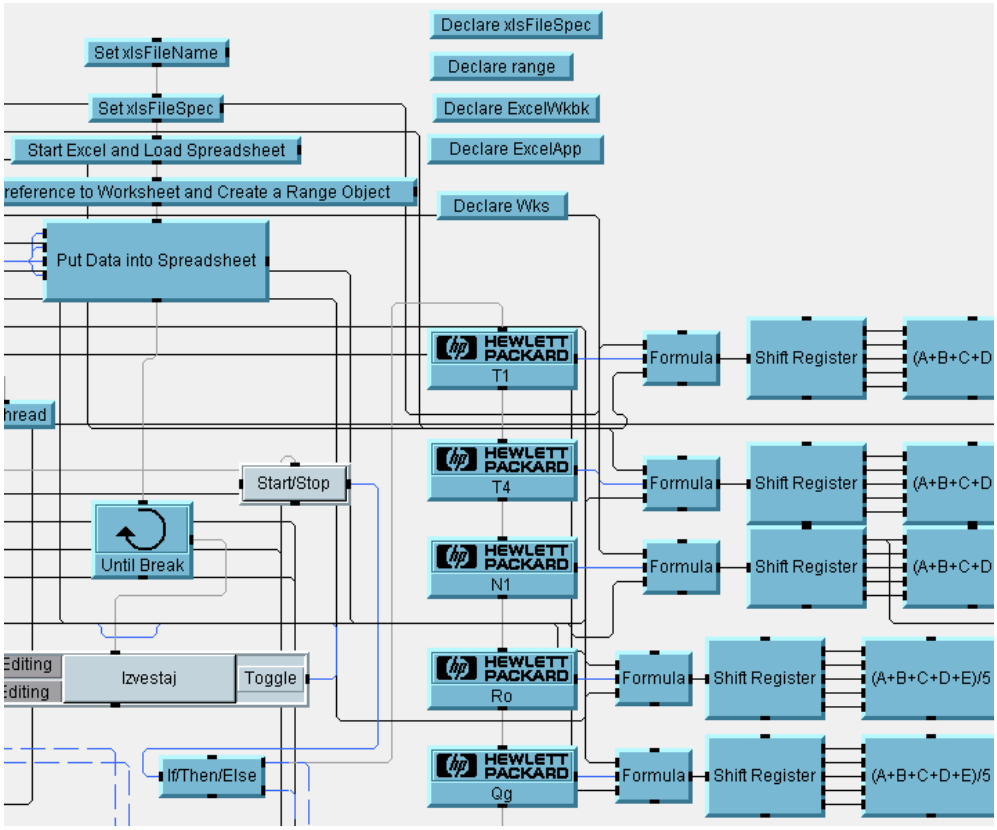
Zbog potrebe da se istovremeno precizno mjeri veliki broj mjernih signala na ispitnoj stanici, predviđena je ugradnja merno-akvizicionog sistema kako bi se povećala brzina i pouzdanost, a smanjila mjerna nesigurnost mjerenja. Mjerno-akvizicioni sistem (u daljem tekstu akvizicioni sistem) povezuje sve mjerne instrumente na ispitnoj stanici u jednu cjelinu koju nazivamo mjerni sistem. Mjerni podaci prikupljeni pomoću akvizicionog sistema treba da se obrade, prikažu i memorišu. Izmjerene vrijednosti mogu se prikazati na ekranu računara i štampati na štampaču. U toku ispitivanja TMM-a na ispitnoj stanici, svi izmjereni parametri motora, snimljeni u realnim atmosferskim uslovima, potrebno je da se u što kraćem vremenu preračunaju (koriguju) na vrijednosti pri standardnim atmosferskim uslovima (temperatura vazduha 15 ^{o}C i atmosferski pritisak 1013 mbar), bez obzira na realne uslove pri snimanju parametara. Suština korekcije izmjerenih parametara motora na standardne atmosferske uslove leži u tome da se omogući poređenje osnovnih radnih parametara motora sa parametrima koje je propisao proizvođač, a koji su dati za standardne atmosferske uslove. Upotrebom akvizicionog sistema prilikom ispitivanja TMM-a na ispitnoj stanici, korigovani parametri se dobijaju u realnom vremenu dok traje ispitivanje motora. Na osnovu izmjerenih i proračunatih korigovanih parametara može se odmah ocjeniti da li je remont dobro obavljen ili je potrebno izvršiti dodatna podešavanja na motoru. U slučaju da korigovani parametri motora malo odstupaju od granica koje je propisao proizvođač, mogu se izvršiti neka manja podešavanja dok motor radi. U slučaju da korigovani parametri motora puno odstupaju od granica koje je propisao proizvođač, treba prekinuti ispitivanje, dok se ne otkloni kvar na motoru, da se ne bi nepotrebno trošilo gorivo za ispitivanje na ostalim radnim režimima. Sva ova snimanja na ispitnoj stanici rade se sa ciljem da se otkrije i otkloni svaki eventualni problem u radu motora na zemlji, a prije njegove ugradnje u avion. Kada se motor ugradi u avion, isti mora da zadovolji izuzetno veliku pouzdanost u radu, u svim režimima u kojima motor može da se nađe prilikom leta aviona.

Radom akvizicionog sistema upravlja korisnički program koji se izvršava na personalnom računaru. Program određuje vrste mjerenja koje treba obaviti i njihov vremenski raspored, oblik i vremenski raspored mjernih podataka koje daju mjerni instrumenti, kao i vrste matematičkih operacija koje izvodi računar nad tim podacima, prije njihovog alfanumjeričkog ili grafičkog predstavljanja. Korisnički

program za akviziciju parametara TMM-a na ispitnoj stanici sastoji se od dva dijela:

- glavnog programa i
- uslužnih programa.

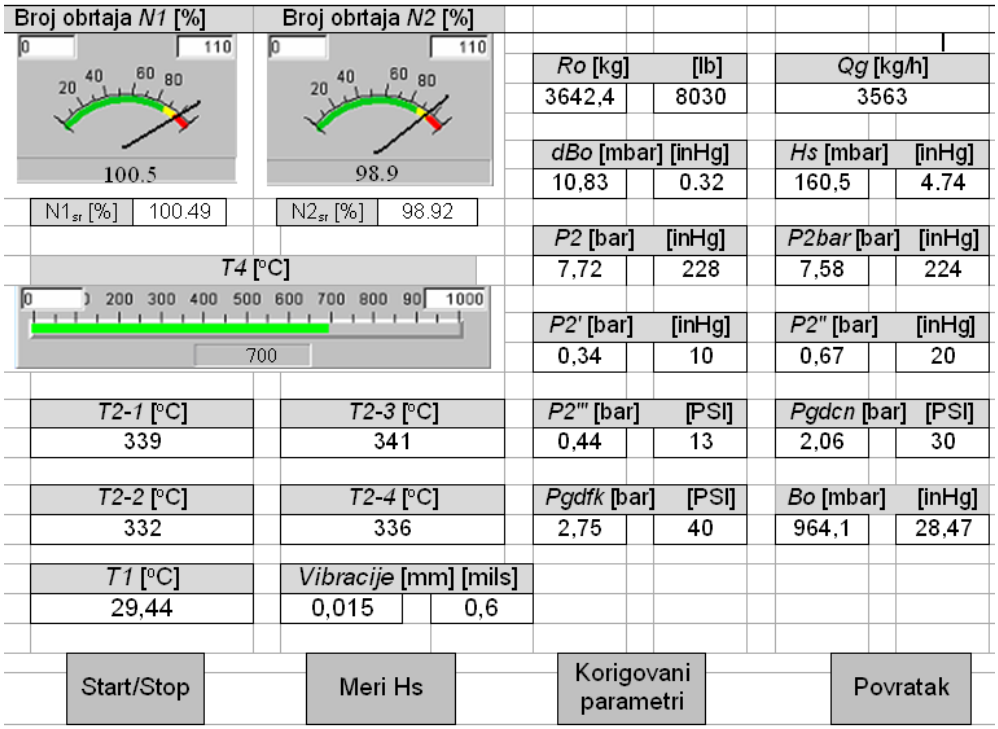
Glavni program je dio korisničkog programa koji opisuje operativni proces sistema i algoritam mjernog ciklusa. Glavni program napisan je u programskom jeziku *Visual Basic* 6.0. Glavni program određuje trenutak startovanja i zaustavljanja mjernog procesa, redoslijed mjernih ciklusa, broj mjerenja u jednom ciklusu i odredište mjernih podataka. U njemu se obavljaju matematičke operacije nad izmjerenim parametrima, kako bi se dobili korigovani parametri i automatski napravio izvještaj o ispitivanju motora. Iz glavnog programa se pozivaju uslužni programi napisani u programskom jeziku *VEE Pro*, komunicira se sa bazama podataka koje su napravljene u *Microsoft Access*-u i komunicira sa *Microsoft Excel* dokumentima koji služe za prikaz krivih motora.



Slika 6.12 Izgled dijela programa za mjerenje parametara motora napisanog u VEE Pro

Uslužni programi sastoje se od rutina (podprograma, procedura, raznih biblioteka) koje se odnose na programiranje i saobraćaj mjernih podataka od računara prema akviziciono-kontrolnoj jedinici HP3852A i od akviziciono-kontrolne jedinice prema računaru. Uslužni programi su pisani u programskom paketu *VEE Pro*. Program se pravi tako što se sastavlja iz gotovih blokova. Svaki blok predstavlja niz naredbi koje čine jednu celinu (n.p. *IF, FOR, UNTIL BREAK* petlje, matematičke funkcije, brojači, grafici, displeji, dugmad i dr.). Programer ove gotove blokove prilagođava svome programu i povezuje ih u jednu cjelinu. Na slici 6.12 prikazan je izgled dijela programa napisanog u programskom paketu *VEE Pro* 6.0. za mjerenje pomoću akviziciono-kontrolne jedinice HP3852A. Primjeri koda programa koji omogućuju komunikaciju računara sa akviziciono-kontrolnom jedinicom HP3852A, preko posebnog bloka naredbi napisanih u programu *VEE Pro* prikazani su ranije na slici 6.5 i slici 6.6.

Nakon pokretanja uslužnog programa, u njega se iz baze podataka automatski unose koeficijenti dobijeni prilikom etaloniranja svakog mjernog signala, a koji se mjere u odabranom režimimu rada motora. Nakon toga, pojavi se prozor prikazan na slici 6.13, preko koga ispitivači vrše upravljanje tokom ispitivanja i prate izmjerene parametre motora.



Slika 6.13 Izgled ekrana prilikom mjerenja parametara motora

Na ovom prozoru vidimo, da se broj obrtaja motora *N1* i *N2* prikazuje u vidu simulacije analognog instrumenta sa kazaljkom i u vidu digitalne vrijednosti. Ovo je urađeno da bi ova dva parametra bili vizuelno što uočljiviji, jer su oni najbitniji za ocjenu u kom radnom režimu se nalazi motor. Ispod ova dva displeja nalazi se displej *N1sr* i *N2sr* koji uvijek prikazuje srednju vrijednost zadnjih deset mjerenja broja obrtaja motora *N1* odnosno *N2,* koja su izvršena akvizicionim sistemom. Ova dva displeja sa srednjom vrednošću zadnjih deset mjerenja broja obrtaja motora *N1* i *N2*, dodati su da bi se poređenjem sa displejima koji stalno pokazuje trenutnu (zadnju) vrednost broja obrtaja motora, moglo utvrditi da li je broj obrtaja motora stabilan, odnosno da je rad motora stabilan. Ako nema velikog odstupanja u pokazivanju broja obrtaja motora na ova dva displeja, može se reći da motor stabilno radi i da se može pristupiti mjerenju ostalih parametara potrebnih za proračun korigovanih parametara motora. Displej za prikaz srednje vrijednosti zadnjih deset mjerenja broja obrtaja motora napravljen je pomoću jednog *shift* registra. *Shift* registar je gotov programski blok koji postoji u programskom paketu *VEE Pro*. Ovaj registar je napravljen tako da u svakom trenutku pamti zadnjih deset (ili već koliko se programira) rezultata mjerenja, koja dolaze na njegov ulaz. Na izlazu iz ovog registra stalno se računa srednja vrijednost od deset vrijednosti, koje se u tom trenutku nalaze u registru. Ovakav registar postoji za svih dvadeset dva signala koji se mjere ovim programom. Tako da se kao izlazni podaci iz ovog programa, odnosno ulazni podaci u program za proračun korigovanih parametara, uzimaju srednje vrednosti zadnjih deset mjerenja za svaki signal koji se mjeri.

Temperatura izduvnih gasova *T4* prikazuje se u vidu položenog bar-grafa i u vidu digitalne vrednosti. Ovo je urađeno da bi ovaj parametar bio, takođe, vizuelno što uočljiviji. Jer ako ova temperatura pređe određene granice koje je deklarisao proizvođač, može doći do trajnog oštećenja motora. Zato operator za ispitivanje motora treba lako da očitava ovu temperaturu i na vrijeme uoči eventualno prekoračenje temperature i preduzme sve potrebne mjere kako bi se ova temperatura brzo smanjila. Smanjivanje ove temperature se radi postepenim smanjivanjem broja obrtaja motora, pošto bi naglo smanjivanje broja obrtaja motora ili naglo zaustavljanje motora moglo takođe da izazove trajno oštećenje motora. Bar-graf je programiran da se ova temperatura prikazuje zelenom bojom sve dok je u granicama koje je deklarisao proizvođač motora. Čim pređe deklarisane granice, ova temperatura se prikazuje crvenom bojom. To je jasan znak operatoru za ispitivanje motora da nešto nije u redu i da treba brzo da preduzme odgovarajuće aktivnosti kako nebi došlo do oštećenja motora.

Potisak motora *Ro* prikazuje se u zakonskoj mjernoj jedinici kg, ali i u librama (1 lb = 0,45359 kg), t.j. u jedinici u kojoj je pokazivanje digitalnog instrumenta na komandno upravljačkom pultu.

Pritisci *Bo, dBo* i *Hs* prikazuju se u zakonskoj mjernoj jedinici mbar, ali i u inčima žive (1 inHg = 33,8639 mbar), t.j. u jedinici u kojoj je pokazivanje vodenih stubova na komandno upravljačkom pultu.

Pritisci *P2, P2bar, P2′* i *P2″* prikazuju se u zakonskoj mjernoj jedinici bar, ali i u inčima žive (1 inHg = 0,0338639 bar), t.j. u jedinici u kojoj je pokazivanje manometara na komandno upravljačkom pultu.

Pritisci *P2‴, Pgdcn* i *Pgdfk* prikazuju se u zakonskoj mjernoj jedinici bar, ali i u PSI (1 PSI = 0,068947 bar), t.j. u jedinici u kojoj je pokazivanje manometara na komandno upravljačkom pultu.

Kod vibracija na motoru mjeri se pomjeraj u zakonskoj jedinici mm i u milsima (1 mils = 0,0254 mm), t.j. u jedinici u kojoj je pokazivanje instrumenta na komandno upravljačkom pultu.

Izmjerene vrednosti u zakonskim jedinicama prikazane su na lijevom displeju, a na desnom displeju su prikazane izmjerene vrijednosti u jedinici u kojoj je pokazivanje instrumenta na komandno upravljačkom pultu.

Postojanje dvije mjerne jedinice za jedan mjerni signal opravdava se obavezom prikaza izmjerenih vrednosti u zakonskim jedinicama *SI* sistema, sa jedne strane, i poređenje pokazivanja akvizicionog sistema i instrumenata na komandno upravljačkom pultu, sa druge strane. Čim se uoči razlika u pokazivanju akvizicionog sistema i instrumenta na komandno upravljačkom pultu potrebno je uraditi vanredno etaloniranje tog mernog signala.

Mjerenje parametara motora pomoću akvizicionog sistema počinje klikom miša na dugme **Start/Stop** (videti sliku 6.13). Mjerenje se ponavlja svake sekunde sve dok se ne zaustavi ponovnim klikom miša na dugme **Start/Stop**. Kada je pritisnuto dugme **Start/Stop**, odnosno dok traje mjerenje akvizicionim sistemom, nije moguć pristup dugmetu **Korigovani Parametri**. Akvizicioni sistem mjeri prvo 14 naponskih signala (*T1, N1, N2, Ro, Qg, Bo, P2, P2bar, P2′, P2″, P2‴, Pgdcn, Pgdfk* i *Vibracije*) koji se dobijaju iz odgovarajućih mjernih pretvarača. Naponski signal se u programu preračunava u odgovarajuću brojnu vrijednost i ta vrijednost se prikazuje na displeju za svaki signal. Zatim akvizicioni sistem mjeri 5 temperatura (*T2-1, T2-2, T2-3, T2-4* i *T4*) pomoću termoparova K-tip i te vrijednosti se prikazuju na posebnom displeju za svaku temperaturu.

Ako želimo da mjerimo pritisak *Hs* onda je potrebno da se pritisne dugme **Mjeri HS**. Kada se pritisne ovo dugme preko relejne kartice akvizicionog sistema uključuju se elektropneumatski ventili koji prvo propuštaju pritisak *Hs1* do davača pritiska. Potrebno je sačekati dvije sekunde da se ovaj pritisak stabilizuje u instalaciji, pa se onda akvizicionim sistemom vrši mjerenje naponskog signala sa davača pritiska. Zatim se preko relejne kartice akvizicionog sistema uključuju elektropneumatski ventili koji propuštaju pritisak *Hs2* do davača pritiska. Potrebno je sačekati dvije sekunde da se ovaj pritisak stabilizuje u instalaciji, pa se onda akvizicionim sistemom vrši mjerenje naponskog signala sa davača pritiska. Dok traje mjerenje pritisaka *Hs1* i *Hs2* ostali signali se ne mjere. Na displeju za pritisak *Hs* prikazuje se srednja vrednost od dva izmjerena pritiska *Hs1* i *Hs2*

$$Hs = \frac{(Hs1 + Hs2)}{2} .$$

Zatim se preko relejne kartice akvizicionog sistema uključuju elektropneumatski ventili koji propuštaju pritisak *dBo* do davača pritiska. Tada je merenje pritiska *Hs* završeno i ponovo počinje merenje ostalih 20 signala. Da bi ponovo izmjerili pritisak *Hs* potrebno je ponovo pritisnuti dugme **Mjeri HS**.

Kada se zaustavi mjerenje (klikom miša na dugme **Start/Stop**) tek onda može da se pritisne dugme **Korigovani Parametri.** Kada se pritisne dugme **Korigovani Parametri** vrijednosti izmjerene akvizicionim sistemom šalju se u glavni program gde se sa njima radi proračun korigovanih parametara motora za izabrani radni režim. Kada se završi proračun korigovanih parametara na ekranu se pojavljuje forma na kojoj se nalaze svi izmjereni parametri i proračunati korigovani parametri TMM-a za izabrani radni režim. Ovi dobijeni parametri se mogu štampati i spašavati u bazu podataka, kako bi se naknadno mogli detaljno analizirati.

Sve vrijednosti izmjerenih i korigovanih parametara motora koji izlaze iz dozvoljenih granica, koje je propisao proizvođač motora, biće obojene crvenom bojom. Ovo u mnogome olakšava rad ispitivaču motora jer odmah lako uočava sve parametre koji su izvan granica koje je propisao proizvođač. On odmah može da donese odluku da li da prekida ispitivanje motora ili da pokuša da uradi manja podešavanja na motoru dok motor radi. Ako su vršena podešavanja na motoru, ispitivač ponavlja mjerenje i računanje korigovanih parametara nekoliko puta da bi se uvjerio da su svi izmjereni i korigovani parametri motora u dozvoljenim ganicama ili treba uraditi dodatna podešavanja na motoru.

Ako se ne želi raditi proračun korigovanih parametara motora sa izmjerenim vrijednostima, onda je potrebno klikom miša pritisnuti dugme **Povratak**, nakon čega se prekida mjerenje, zatvara uslužni program i vraćamo se u glavni program gdje se bira režim rada motora.

Glavni doprinos realizacije i primjene akvizicionog sistema na ispitnoj stanici za ispitivanje TMM-a je unapređenje procesa ispitivanja i smanjenje vremena ispitivanja. Smanjenjem vremena ispitivanja smanjuje se utrošak goriva, ulja i električne energije. Sve ovo dovodi do smanjenja cijene ispitivanja TMM-a na ispitnoj stanici. Primjena akvizicionih sistema u ispitivanju turbo-mlaznih motora je veoma značajan rezultat, jer tržište danas konstantno zahteva umanjenje cijene ispitivanja, ali ujedno i poboljšanje kvaliteta ispitivanja.

7 SIMULACIJA LABORATORIJSKIH VJEŽBI NA RAČUNARU

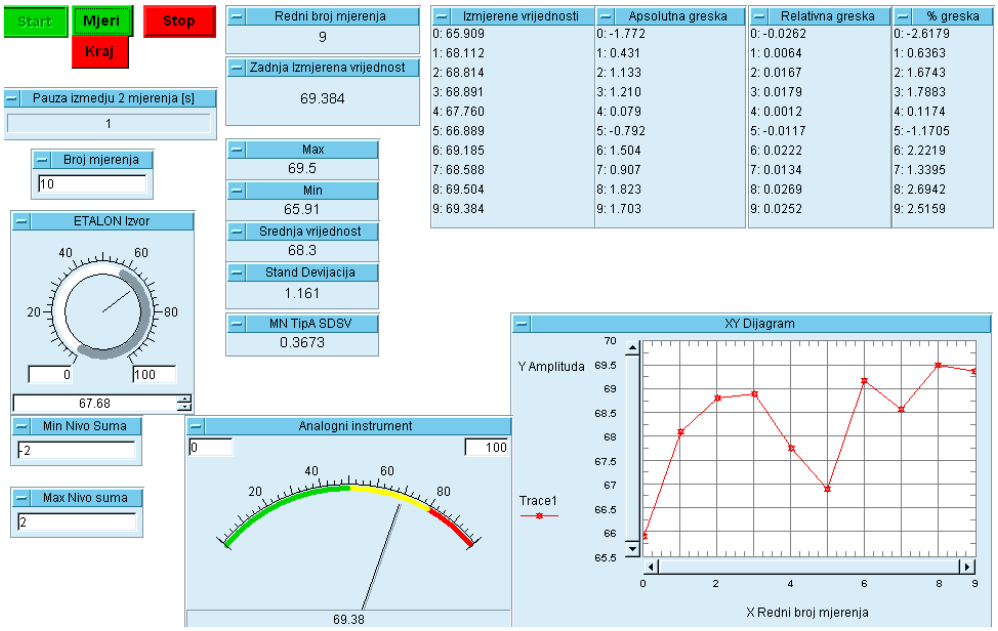
Programski jezik *VEE Pro* može da posluži za pisanje programa, koji predstavljaju simulaciju na računaru laboratorijskih vježbi iz predmeta Električna mjerenja. Simulacijom laboratorijskih vježbi na računaru studenti se na predavanjima i kroz samostalan rad pripremaju za praktične laboratorijske vježbe. Ovo je posebno značajno za studente, koji se prvi put sreću sa mjernim instrumentima. Simulacijom mjernih instrumenata na računaru studentima se mogu predstaviti sve opasnosti koje prijete mjernim instrumentima, ali i osobama koje vrše mjerenje. Te opasnosti najčešće proizilaze zbog nedovoljnog poznavanja mjernih instrumenata i mjernih matoda koje su predmet laboratorijskih vježbi. Ovo posebno dolazi do izražaja kada su u istom trenutku u laboratoriji postavljene različite vježbe, a što je uslovljeno nedovoljnim brojem istih mjernih instrumenata. Sa druge strane na računaru se istovremeno mogu postaviti razne laboratorijske vježbe, a studenti se mogu za njih pripremati i kod kuće na svom računaru. Na ovaj način se praktično rad laboratorije može povećati na 24 sata, sa velikim brojem radnih mjesta.

Iz dosadašnjeg iskustva u radu sa studentima, pokazalo se da i pored teoretskih priprema za izvođenje laboratorijskih vježbi, u praksi često dolazi do kvarova na mjernim instrumentima. Ovi kvarovi obično predstavljaju trajna fizička oštećenja mjernih instrumenata i mjerne opreme. Na taj način nastaju problemi u daljem izvođenju laboratorijskih vježbi, kao i velike materijalne štete za fakultet. Simulacijom laboratorijskih vježbi na računaru studenti će se moći puno bolje pripremiti za praktična mjerenja, a kvarovi na laboratorijskim mjernim instrumentima prouzrokovani ljudskim faktorom treba da se smanje. Na računaru se pomoću programskog jezika mogu simulirati i neka mjerenja, koja nije moguće obaviti u laboratoriji na fakultetu u realnim uslovima. Generatori funkcija, generatori impulsa, generatori šuma, kalibratori, osciloskopi, spektralni analizatori i digitalni multimetri su izuzetno skupi mjerni instrumenti, da bi se u laboratoriji za studente postavilo 10 radnih mjesta sa svim ovim mjernim instrumentima. U programskom jeziku *VEE Pro* postoje gotovi objekti, koji vjerno simuliraju ove mjerne instrumente. Pomoću ovih objekata moguće je simulirati laboratorijske vježbe i iz drugih predmeta na Elektrotehničkom fakultetu. Na ovaj način se kvalitet nastave iz predmeta Električna mjerenja, ali i drugih predmeta podiže na jedan viši nivo.

Takođe ćemo nastojati stimulisati studente da i oni sami prave nove laboratorijske vježbe u programskom jeziku *VEE Pro* kroz projekte i diplomske radove.

7.1 OBRADA REZULTATA MJERENJA

Prvo će biti predstavljena simulacija laboratorijske vježbe za statističku obradu rezultata mjerenja. Izgled izvršne verzije ovog programa prikazan je na slici 7.1.



Slika 7.1 Obrada rezultata mjerenja

Nakon pokretanja programa, a prije otpočinjanja mjerenja potrebno je unijeti početne parametre za svako mjerenje. Preko objekta za unos konstanti prvo se bira vrijeme u sekundama, između dva mjerenja. Ako se rezultati mjerenja žele pojedinačno ručno zapisivati onda je neophodno da ova vrijednost bude veća od 10 sekundi. Odnosno svaki operater treba da procjeni vrijeme koje mu je potrebno da izvrši očitanje i da izmjerenu vrijednost zapiše na papiru. Ako se mjerenje želi ubrzati, tako da se rezultati mjerenja zapisuju samo na računaru. U tom slučaju se ova vrijednost može postaviti na jednu sekundu ili manje. Zatim se unosi broj mjerenja za svaku mjernu tačku. Promjenom ove vrijednosti može se pokazati da se ukupna greška mjerenja i standardna devijacija smanjuje sa povećanjem broja mjernih tačaka. Amplituda signala, čije se mjerenje simulira, postavlja se preko virtuelnog izvora (koji na slici 7.1 nosi naziv ETALON izvor). Nova vrijednost se postavlja mišem kontinualno preko kružnog potenciometra, ili direktnim upisivanjem brojne vrijednosti u digitalni displej, koji se nalazi na dnu objekta. Kako bi se dobio mjerni signal sa promjenljivom amplitudom (koja se mijenja po nepoznatom zakonu) na ovu izabranu vrijednost signala se dodaje šum iz generatora slučajnih brojeva. Nivo šuma se takođe može mijenjati preko dva

objekta, u koja se unosi minimalna vrijednost amplitude šuma (Min Nivo Suma) i maksimalna vrijednost amplitude šuma (Max Nivo Suma). Minimalnu vrijednost amplitude šuma potrebno je unijeti sa predznakom minus. Ovom laboratorijskom vježbom se pokazuje da nivo šuma utiče na vrijednost standardne devijacije mjerenja. Eksperimentalno se može pokazati da povećanje broja mjernih tačaka smanjuje uticaj šuma na srednju vrijednost signala. Za svako mjerenje automatski se računa apsolutna, relativna i procentualna greška mjerenja. Takođe se automatski vrši i statistička obrada rezultata mjerenja računanjem: srednje, maksimalne i minimalne vrijednosti, standardne devijacije i standardne devijacija srednje vrijednosti. Standardna devijacija srednje vrijednosti ujedno predstavlja i mjernu nesigurnost tip A za izvršenu seriju mjerenja. Svaka izmjerena vrijednost se prikazuje u vidu brojne vrijednosti (digitalni mjerni instrument), na XY dijagramu i na objektu, koji predstavlja simulaciju analognog mjernog instrumenta. XY dijagram se koristi da bi studenti lakše uočili rasipanje rezultata mjerenja oko srednje vrijednosti. Simulaciju analognog mjernog instrumenta koristimo, kako bi se uočilo lakše i preciznije očitanje izmjerenih vrijednosti na digitalnom displeju, u odnosu na očitanje na analognom instrumentu. Međutim, ovo vrijedi samo ako je mjerna veličina sporo promjenljiva. Ako je mjerna veličina brzo promjenljiva lakše je vršiti očitanje na analognom, nego na digitalnom instrumentu. Kada se mjerna veličina brzo mijenja, tada je praktično gotovo nemoguće izvršiti očitanje na digitalnom instrumentu, jer se brojne vrijednosti na digitalnom displeju stalno mijenjaju. Sa druge strane kazaljka analognog mjernog instrumenta vjerno prati promjene mjerne veličine. To je i razlog što su se analogni instrumenti još uvijek zadržali u laboratorijama, ali i u praksi. Brzinomjeri i mjerači obrtaja motpora su i u najnovijim automobilima i avionima još uvijek analogni. Međutim, kod mjernih veličina koje se sporo mijenjaju, pomoću digitalnih instrumenata, očitanje se može izvršiti puno lakše i preciznije. Preciznost mjerenja pomoću digitalnih mjernih instrumenata zavisi od broja digita koliko mjerni instrument ima. Kod digitalnih mjernih instrumenata manja rezolucija znači i veća brzina mjerenja, a veća rezolucija automatski znači i sporija mjerenja sa istim instrumentom. U ovoj laboratorijskoj vježbi podešavanjem vremena između dva mjerenja može se uočiti kada je lakše očitanje na analognom, a kada na digitalnom instrumentu.

Rezultati mjerenja upisuju se u tabelu 7.1. Prvo je potrebno upisati vrijednost postavljenu na etalonskom izvoru u polje "*TV*". Ova vrijednost ostaje nepromijenjena u ovom nizu mjerenja. Prilikom statističke obrade rezultata mjerenja ova vrijednost se uzima kao tačna vrijednost. Apsolutna, relativna i procentualna greška za svako mjerenje računa se upravo u odnosu na ovu vrijednost. Zatim se upisuje minimalna i maksimalna vrijednost amplitude šuma u toku ovog mjerenja u polja "*MinŠum*" i "*MaxŠum*". U polje "*n*" potrebno je unijeti broj mjernih tačaka, jer ovaj broj direktno učestvuje u formuli za računanje srednje vrijednosti i standardne devijacije. Nakon završenog mjerenja u ovu tabelu se upisuju i statistički obrađeni rezultati mjerenja: apsolutna, relativna, procentualna i

prividna greška za svako mjerenje. Tabelu 7.1 prvo popuniti sa nizom od 10 mjerenja, a zatim istu tabelu popuniti sa nizom od novih 20 mjerenja. Prilikom popunjavanja ove dvije tabele, potrebno je zadržati istu postavljenu vrijednost na etalonskom izvoru, kao i vrijednosti amplitude šuma. Izmjerene vrijednosti se čitaju na objektu, koji predstavlja simulaciju digitalnog mjernog instrumenta (objekat pod nazivom "Zadnja izmjerena vrijednost").

Tabela 7.1 se može popuniti i sa vrijednostima koje se dobiju mjerenjem na objektu, koji predstavlja simulaciju analognog mjernog instrumenta, ali onda to treba posebno naglasiti u nazivu tabele.

Tabela 7.1 Rezultati mjerenja

TV=		*MinŠum=*		*MaxŠum=*		*n=*
i	Izmjerena vrijednost x_i	Apsolutna greška $x_i - TV$	Relativna greška $(x_i - TV)/TV$	Procentualna greška $RG*100$	Prividna greška $(x_i - x_s)$	$(x_i - x_s)^2$
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
$x_s =$						
$s =$						
$s_{x_s} =$						

Srednja vrijednost mjerenja x_s računa se prema formuli (7.1), koja po pravilu predstavlja krajnju vrijednost mjerenja

$$x_s = \frac{1}{n}\sum_{i=1}^{n} x_i \ . \tag{7.1}$$

Standardno odstupanje pojedinih rezultata računa se izrazom (7.2),

$$s = \sqrt{\frac{\sum_{i}^{n}(x_i - x_s)^2}{n-1}} \ . \tag{7.2}$$

Standardno odstupanje srednje vrijednosti predstavlja standardnu nesigurnost tip A mjernog rezultata i računa se izrazom (7.3)

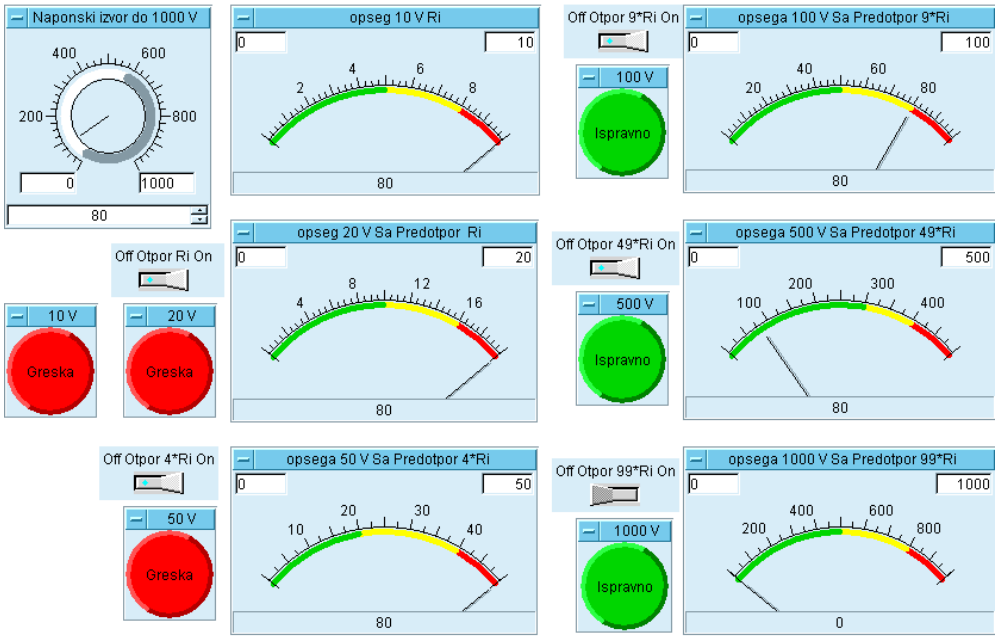
$$u_{Ax_s} = s_{x_s} = \frac{s}{\sqrt{n}} = \sqrt{\frac{\sum_{i}^{n}(x_i - x_s)^2}{n(n-1)}} \tag{7.3}$$

7.2 PROŠIRENJE MJERNOG OPSEGA VOLTMETRA

Sada će biti predstavljena simulacija laboratorijske vježbe za proširenje mjernog opsega voltmetra. Ova simulacija je urađena, kako bi se studenti upoznali sa načinom proširenja mjernog opsega voltmetra dodavanjem u seriju sa voltmetrom otpora velike vrijednosti, koja treba biti veća od unutrašnjeg otpora voltmetra. Posebno se želi pokazati značaj izbora odgovarajućeg mjernog opsega, kako ne bi došlo do trajnog oštećenja voltmetra, kada kazaljka prekorači vrijednost punog mjernog opsega. Izgled izvršne verzije ovog programa prikazan je na slici 7.2.

Amplituda naponskog signala, čije se mjerenje simulira, se postavlja preko virtuelnog izvora (koji može generisati maksimalni napon od 1000 V). Željena vrijednost se postavlja mišem preko kružnog potenciometra, ili direktnim upisivanjem brojne vrijednosti u digitalni displej, koji se nalazi na dnu objekta. U programu postoji simulacija šest analognih voltmetara sa šest različitih mjernih opsega (10 V, 20 V, 50 V, 100 V, 500 V i 1000 V). Samo voltmetar sa mjernim opsegom od 10 V direktno mjeri napon i nema mogućnost dodavanja predotpora. Svi ostali voltmetri mjere napon tek kada im se priključi predotpor preko prekidača "*On Off*" na kome stoji upisana vrijednost predotpora od *Ri* do 99**Ri*, pri čemu je

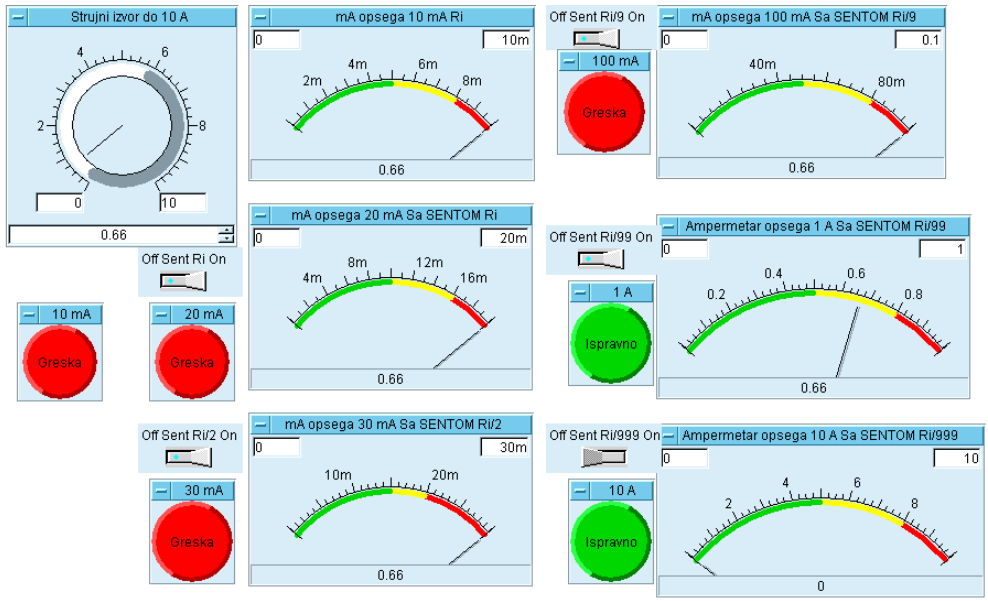
Ri unutrašnja otpornost voltmetra na osnovnom mjernom opsegu od 10 V. Na ovaj način se dodavanjem predotpora mjerni opseg voltmetra povećava od 2 do 100 puta. Ispravno mjerenje napona se signalizira sa zelenim poljem u kome se pojavi poruka "Ispravno" uz taj instrument. Kada se pokuša mjeriti napon koji je veći od mjernog opsega instrumenta, automatski se izvrši signalizacija sa crvenim poljem u kome se pojavi poruka "Greška" uz taj instrument. Kod realnih instrumenata dovođenje napona, koji je veći od mjernog opsega bi dovelo do oštećenja tog mjernog opsega ili čak do potpunog uništenja instrumenta.



Slika 7.2 Proširenje mjernog opsega voltmetra

7.3 PROŠIRENJE MJERNOG OPSEGA AMPERMETRA

Sada će biti predstavljena simulacija laboratorijske vježbe za proširenje mjernog opsega ampermetra. Ova simulacija je urađena, kako bi se studenti upoznali sa načinom proširenja mjernog opsega ampermetra dodavanjem u paralelu sa instrumentom otpora male vrijednosti, koja treba biti manja od unutrašnjeg otpora ampermetra. Ovaj otpor se zove šent. Posebno se želi pokazati značaj izbora odgovarajućeg mjernog opsega, kako ne bi došlo do trajnog oštećenja ampermetra kada kazaljka prekorači vrijednost punog mjernog opsega. Izgled izvršne verzije ovog programa prikazan je na slici 7.3.



Slika 7.3 Proširenje mjernog opsega ampermetra

Amplituda strujnog signala čije se mjerenje simulira postavlja se preko virtuelnog izvora (koji može generisati maksimalnu struju od 10 A). Željena vrijednost se postavlja mišem preko kružnog potenciometra, ili direktnim upisivanjem brojne vrijednosti u digitalni displej, koji se nalazi na dnu objekta. U programu postoji simulacija šest analognih ampermetra sa šest različitih mjernih opsega (10 mA, 20 mA, 30 mA, 100 mA, 1 A i 10 A). Samo ampermetar sa mjernim opsegom od 10 mA direktno mjeri struju i nema mogućnost dodavanja šenta. Svi ostali ampermetri mjere struju tek, kada im se priključi šent preko prekidača "*On Off*" na kome stoji upisana vrijednost otpora šenta od *Ri* do *Ri*/999, pri čemu je *Ri* unutrašnja otpornost ampermetra na osnovnom mjernom opsegu od 10 mA. Na ovaj način se dodavanjem šenta mjerni opseg ampermetra povećava od 2 do 100 puta. Ispravno mjerenje struje se signalizira sa zelenim poljem u kome se pojavi poruka "Ispravno" uz taj instrument. Kada se pokuša mjeriti struja koja je veća od mjernog opsega instrumenta, automatski se izvrši signalizacija sa crvenim poljem u kome se pojavi poruka "Greška" uz taj instrument. Kod realnih instrumenata dovođenje struje, koja je veća od mjernog opsega bi dovelo do oštećenja tog mjernog opsega ili čak do potpunog uništenja instrumenta.

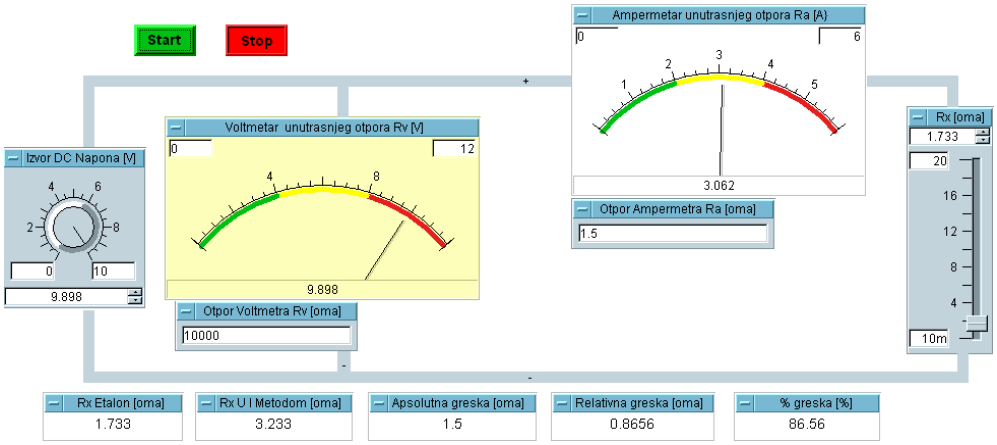
7.4 MJERENJE OTPORA UI METODOM

Simulacija laboratorijske vježbe za mjerenje nepoznatog otpora pomoću UI metode prikazana je na slici 7.4 i 7.5. Ova simulacija je urađena, kako bi se studenti upoznali sa greškom mjerenja otpora korišćenjem UI metode. Greška mjerenja otpora nastaje jer realni ampermetar ima konačnu unutrašnju otpornost, a idealno bi bilo da uopšte nema otpora (odnosno da je otpor $Ra = 0\ \Omega$). Takođe i realni voltmetar ima konačnu unutrašnju otpornost, a idealno bi bilo da ima beskonačan otpor (odnosno da je otpor $Rv = \infty\ \Omega$). Nepoznati otor *Rx* čiju vrijednost želimo da izmjerimo predstavljen je preko simulacije kliznog otpornika. Željena vrijednost otpora se postavlja mišem preko potenciometra, ili direktnim upisivanjem brojne vrijednosti u digitalni displej, koji se nalazi na vrhu objekta. Ova vrijednost se u postupku računanja greške UI metode uzima za tačnu vrijednost (vrijednost etalona). Amplituda DC naponskog signala, koji napaja ovo električno kolo se simulira preko virtuelnog izvora (koji može generisati maksimalni napon od 10 V). Željena vrijednost napona se postavlja mišem preko kružnog potenciometra, ili direktnim upisivanjem brojne vrijednosti u digitalni displej, koji se nalazi na dnu objekta. U polja ispod voltmetra i ampermetra se upisuje vrijednost unutrašnjeg otpora ovih instrumenata. Ovom vježbom se pokazuje da ove vrijednosi direktno utiču na grešku mjerenja otpora UI metodom. Promjenom napona napajanja mijenja se i pokazivanje instrumenata, a samim tim i dobijena vrijednost nepoznatog otpora *Rx* izračunata preko UI metode. Automatski se prikazuje apsolutna, relativna i procentualna greška mjerenja otpora UI metodom. Eksperimentalno se može pokazati da se prilikom mjerenja velikih otpora tačnije mjerenje UI metodom dobije za šemu spajanja prikazanu na slici 7.4. Tada ampermetar direktno mjeri struju, koja teče kroz otpornik čija se otpornost mjeri. Voltmetar mjeri napon na ampermetru i otporniku *Rx*. Kada je unutrašnja otpornost ampermetra mala u odnosu na otpornost nepoznatog otpornika *Rx*, tada je i pad napona na ampermetru zanemariv, pa se može smatrati da voltmetar mjeri napon na otporniku *Rx*. Međutim, do greške mjerenja otpora sa ovom metodom dolazi ipak zbog pada napona na ampermetru, koji ima konačnu vrijednost.

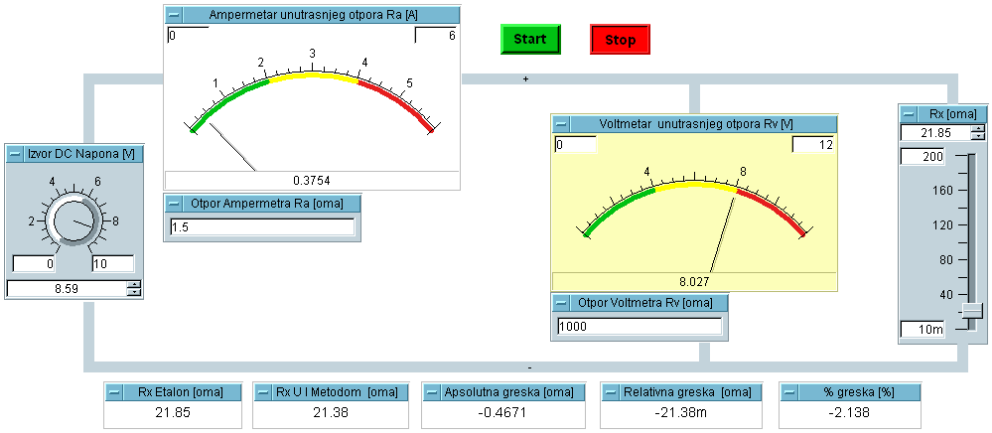
Prilikom mjerenja malih otpora tačnije mjerenje se dobije za šemu spajanja prikazanu na slici 7.5. Tada voltmetar direktno mjeri napon na otporniku čija se otpornost mjeri. Ampermetar tada mjeri struju, koja teče kroz voltmetar i otpornik *Rx*. Kada je unutrašnja otpornost voltmetra velika u odnosu na otpornost nepoznatog otpornika *Rx*, tada je i struja koja teče kroz voltmetar zanemariva, pa se može smatrati da ampermetar mjeri struju kroz otpornik *Rx*. Međutim, do greške mjerenja otpora sa ovom metodom dolazi ipak zbog struje, koja teče kroz voltmetar i koja ima konačnu vrijednost.

Prilikom mjerenja nepoznatih otpora čija je vrijednost veća od unutrašnje otpornosti ampermetra, a manja od unutrašnje otpornosti voltmetra treba biti obazriv prilikom korišćenja UI metode za mjerenje otpora. Tada treba procijeniti,

koja od ove dvije šeme spajanja daje manju grešku mjerenja. Prilikom izvođenja ove laboratorijske vježbe nepoznati otpor potrebno je izmjeriti prvo sa jednom, a zatim i sa drugom šemom spajanja.



Slika 7.4 UI metoda mjerenja otpora, ampermetar mjeri direktno stuju kroz Rx



Slika 7.5 UI metoda mjerenja otpora, voltmetar mjeri direktno napon na Rx

Rezultati mjerenja upisuju se u tabelu 7.2 pri čemu kolone imaju sljedeća značenja:

- *Rt* predstavlja vrijednost otpora koja se postavlja na kliznom otporniku *R*x.
- *Rv* predstavlja unutrašnju otpornost voltmetra.
- *Ui* predstavlja vrijednost napona izmjerenu na voltmetru.
- *R*a predstavlja unutrašnju otpornost ampermetra.
- *Ii* predstavlja vrijednost struje izmjerene na ampermetru.

- $Ri = Ui / Ii$ vrijednost otpora izmjerena UI metodom.
- $\Delta R = Ri - Rt$ predstavlja apsolutnu grešku mjerena otpora UI metodom.
- $\Delta R\% = (Ri - Rt)*100/ Rt$ predstavlja procentualnu grešku mjerena otpora.

Na kraju laboratorijske vježbe potrebno je dati komentar sa kojom šemom spajanja se dobije manja greška mjerenja za sve četiri vrijednosti otpora *R*x.

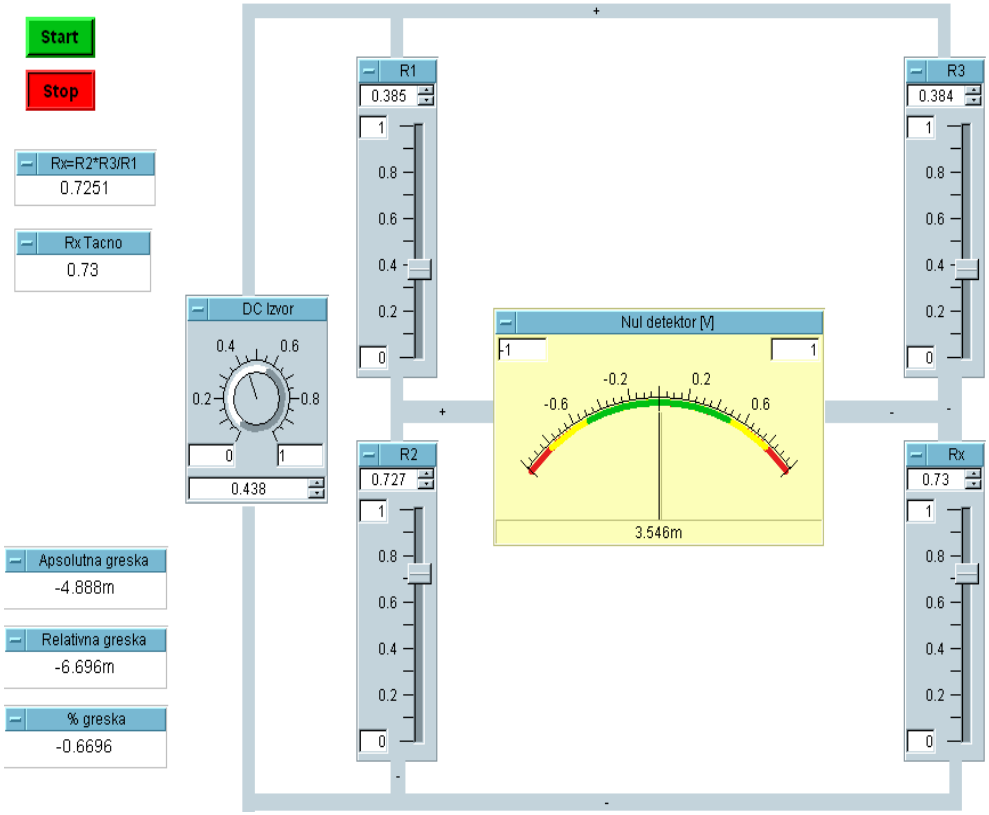
Tabela 7.2 Mjerenje otpora UI metodom, ampermetar mjeri direktno stuju kroz Rx (drugu tabelu popuniti kada voltmetar mjeri direktno napon na Rx)

r.b.	*Rt* [Ω]	*Rv* [kΩ]	*Ui* [V]	*Ra* [Ω]	*Ii* [mA]	*Ri* [Ω]	*ΔR* [Ω]	*ΔR%*
1.	10	50		2				
2.	10	500		2				
3.	10	50		20				
4.	10	500		20				
5.	1000	50		2				
6.	1000	500		2				
7.	1000	50		20				
8.	1000	500		20				
9.	10000	50		2				
10.	10000	500		2				
11.	10000	50		20				
12.	10000	500		20				
13.	100000	50		2				
14.	100000	500		2				
15.	100000	50		20				
16.	100000	500		20				

7.5 VJEŽBA MJERNI MOST

Simulacija laboratorijske vježbe za mjerenje nepoznatog otpora pomoću jednosmjernog Vitstonovog mosta prikazana je na slici 7.6. Ova simulacija je urađena, kako bi se studenti upoznali sa načinom mjerenja u mosnim metodama. Na ovoj električnoj šemi nalazi se simulacija jednosmjernog naponskog izvora, četiri promjenljiva otpornika i nul naponski detektor. Otpornici *R1, R2, R3* i *Rx* predstavljeni su preko simulacije kliznog otpornika. Željena vrijednost otpora se postavlja mišem preko potenciometra, ili direktnim upisivanjem brojne vrijednosti u digitalni displej, koji se nalazi na vrhu objekta. Vrijednost postavljena na otporniku *Rx* se u postupku računanja greške mostnom metodom uzima za tačnu vrijednost (vrijednost etalona). Amplituda DC naponskog signala, koji napaja ovaj

most se simulira preko virtuelnog izvora. Željena vrijednost se postavlja mišem preko kružnog potenciometra, ili direktnim upisivanjem brojne vrijednosti u digitalni displej, koji se nalazi na dnu objekta. Eksperimentalno se može pokazati da promjena ovog napona ne utiče na uravnoteženje mosta, a samim tim ni na tačnost mjerenja otpora ovom metodom.



Slika 7.6 Vitstonov most

Most se dovodi u ravnotežu podešavanjem vrijednosti otpora na otpornicima *R1, R2* i *R3*. Smatra se da je most doveden u ravnotežu kada nul detektor pokazuje najmanju vrijednost. Idealno bi bilo da on pokazuje vrijednost nula, ali praktično se to može rijetko postići. Razlog za ovo leži u ograničenoj rezoluciji promjene otpora na objektima, koji simuliraju klizne otpornike. Kada se most dovede u ravnotežu tada vrijedi

$$\frac{R_1}{R_2} = \frac{R_3}{R_x} \tag{7.4}$$

odnosno nepoznata vrijednost otpora dobijena mostnom metodom se dobije pomoću formule

$$R_x = \frac{R_2 \bullet R_3}{R_1} \tag{7.5}$$

Prilikom dovođenja mosta u ravnotežu, automatski se u svakom trenutku računa apsolutna, relativna i procentualna greška mjerenja mosne metode. Cilj je, naravno, da greška mjerenja ima što manju vrijednost.

Rezultati mjerenja upisuju se u tabelu 7.3 pri čemu kolone imaju sljedeća značenja:

- *R*t predstavlja vrijednost otpora koja se postavlja na kliznom otporniku *R*x.
- *R*1, *R*2 i *R*3 predstavlja vrijednost otpora na otporniku, koji se podešava.
- *Rx*i = *R*2***R*3 / *R*1 vrijednost otpora izmjerena mostnom metodom.
- *U*d predstavlja vrijednost napona izmjerenu na nuldetektoru.
- *ΔR* = *R*i - *R*t predstavlja apsolutnu grešku mjerena otpora.
- *ΔR% = (Rx*i - *R*t)**100/ *R*t predstavlja procentualnu grešku mjerena otpora.

Tabela 7.3 Mjerenje otpora Vitstonovim mostom

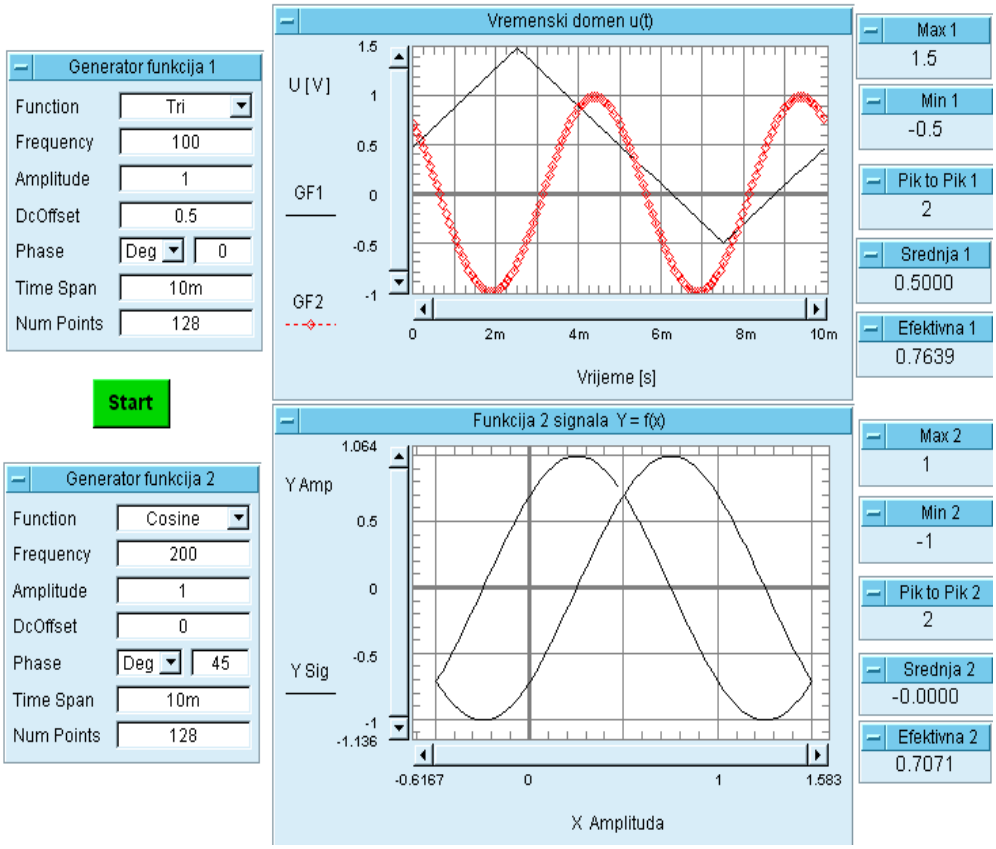
r.b.	*R*t [Ω]	*R*1 [Ω]	*R*2 [Ω]	*R*3 [Ω]	*Rx*i [Ω]	*U*d [V]	*ΔR* [Ω]	*ΔR%*
1.	0,5							
2.	5							
3.	50							
4.	500							

7.6 MJERENJE OSCILOSKOPOM

Simulacija laboratorijske vježbe za mjerenje pomoću osciloskopa prikazana je na slici 7.7. Simulirana su dva virtuelna generatora funkcija, dva osciloskopa i displeji za prikaz maksimalne, minimalne, pik to pik, srednje i efektivne vrijednosti svakog signala na izlazu iz generatora funkcija. Pomoću virtuelnih generatora funkcija, može se mijenjati oblik signala (sinusni, kosinusni, četvrtke, trouglasti i rampa) u polju *Function*, frekvencija signala u polju *Frequency*, amplituda signala u polju *Amplitude*, početno pomjeranje signala po Y osi u polju *DC Offset*, početna fazu signala u polju *Phase,* dužinu vremenske X ose (ovo određuje broj perioda signala koje će biti generisane) u polju *Time Span* i osjetljivost generatora prema broju tačaka (32, 64, 128, ... 2056) koje čine signal u polju *Num Points*. Veći broj tačaka znači i bolji kvalitet signala, odnosno manje izobličenje. Virtualni generatori generišu signal u vidu brojnih vrijednosti zapisanih kao jednodimenzioni niz.

Simulacija prvog osciloskopa se koristi za prikaz vremenskog oblika dva signala iz obadva generatora funkcija na istom ekranu. Na Y osi se prikazuje amplituda signala, a na X osi se prikazuje vrijeme u sekundama (ili mili sekundama). X i Y osa se automatski mijenjaju kako se mijenja signala, koji se snima.

Simulacija drugog osciloskopa se koristi za prikaz međusobne zavisnosti dva signala. Na jedan ulaz osciloskopa se dovodi jedan signal (X osa signal GF1), a na drugi ulaz osciloskopa se dovodi drugi signal (Y osa signal GF2). Ovaj osciloskop se koristi za snimanje Lisažeovih figura. Figure su složenije kada se na ulaz dovode naizmjenični signali različitih frekvencija, koji su pri tome i fazno pomjereni.



Slika 7.7 Simulacija osciloskopa

Na prvom generatoru funkcija (Generator funkcija1) potrebno je postaviti željeni oblik signala, amplitudu i frekvenciju. Signal posmatrati i mjeriti na prvom osciloskopu (Vremenski domen $u(t)$). Potrebno je očitati maksimalnu, efektivnu i

srednju vrijednost signala. Pri tome parametar *DcOffset* treba podesiti na nula, kako bi imali simetričan signal oko nulte vrijednosti. Da bi se na odgovarajućem digitalnom displeju (Srednja1) dobila odgovarajuća srednja vrijednost signala potrebno je na generatoru funkcija podesiti parametar *Time Span* na vrijednost tako da se na osciloskopu dobije samo pozitivna perioda prostoperiodičnog signala. Jer je srednja vrijednost sinusnog signala nula, pa je kod računanja srednje vrijednosti potrebno u proračun uzeti samo jednu polu periodu sinusnog signala. Na primjer, za frekvenciju 100 Hz, potrebno je podesiti parametar *Time Span* na vrijednost 5 mili sekundi. Rezultati mjerenja upisuju se u tabelu 7.4.

Tabela 7.4 Mjerenje osciloskopom

Oblik signala	*U*max [V]	*U*ef [V]	*U*sr [V]	*f* [Hz]	*T* [s]
sinusni					
trouglasti					
četvrtke					

8 PROCJENA MJERNE NESIGURNOSTI

Riječ "nesigurnost" izražava sumnju, a "mjerna nesigurnost", odnosno "nesigurnost mjerenja" izražava sumnju u validnost rezultata mjerenja. Rezultat mjerenja je vrijednost dobijena mjerenjem, koja se pripisuje mjernoj veličini. Potpuno iskazivanje rezultata mjerenja treba obuhvatiti i informaciju o mjernoj nesigurnosti. Nesigurnost rezultata mjerenja je posljedica nedostataka egzaktnog poznavanja vrijednosti koja se mjeri. Rezultat mjerenja poslije korigovanja za prepoznate sistematske efekte je i dalje samo procjena mjerene vrijednosti zbog:

- nesigurnosti koja potiče od slučajnih efekata i
- nesavršenosti korekcije za sistematske efekte.

Postupak mjerenja je složen proces i obuhvata niz elemenata, kao što su informacije o etalonu i mjerilu, metodi mjerenja, korekcijama na dejstvo određenih uticajnih veličina, nesigurnosti sa kojom je dobijen rezultat i slično. Rezultat svakog realnog mjerenja sadrži u sebi određenu nesigurnost, što znači da se idealno tačna vrijednost mjerene veličine ne može saznati. Uzroci mjernih nesigurnosti mogu biti veoma brojni i po pravilu se ne mogu svi uzeti u obzir. Kada se prikazuju rezultati mjerenja, potrebno je jasno iskazati, da li on predstavlja pokazivanje mjerila, korigovanu ili nekorigovanu vrijednost mjerene veličine, i da li je na osnovu niza ponovljenih mjerenja izvršeno usrednjavanje.

Da bi se postigla jednoobraznost u izražavanju mjernih rezultata vodeće institucije međunarodnog metrološkog sistema (*BIPM, IEC, IFCC, ISO, IUPAC, IUPAP* i *OIML*), godine 1993. publikovale su *Uputstvo za izražavanje mjerne nesigurnosti*[8]. Ovo uputstvo definiše harmonizovane i usaglašene metode izračunavanja i izražavanja mjerne nesigurnosti u laboratorijama za etaloniranje i ispitivanje sa procedurom proračuna mjerne nesigurnosti. Do tada za obradu i izražavanje rezultata bila je primjenjivana klasična matematička disciplina, tzv. teorija grešaka. Načini obrade rezultata nisu bili ujednačeni pa je bilo teško poređenje istih mjerenja izvršenih u različitim institucijama.

Obradom mjernih rezultata na način opisan u *Uputstvu* dobija se mjerni rezultat, zatim mjerna nesigurnost i statistička sigurnost sa kojom važe dobijeni podaci. Pojam greška mjerenja odnosi se isključivo na razliku dobijenog rezultata i odgovarajuće vrijednosti dobijene pomoću etalonskog mjernog instrumenta. Koncept izražavanja mjerne nesigurnosti iznesen u Uputstvu ima kao teorijsku podlogu isti matematički aparat, koji se koristi kod teorije grešaka. Međutim, metod izražavanja mjerne nesigurnosti je praktično orijentisan i može se dobro primijeniti u svim eksperimentalnim mjerenjima. Nasuprot tome, polazne veličine

[8] Guide to the Expression of the Uncertainty in Measurement, ISO, 1993

teorije grešaka su slučajne i sistematske greške koje kao idealizovani pojmovi ne postoje u praksi.

Sistematska greška predstavlja razliku između srednje vrijednosti koja bi se dobila iz beskonačnog broja mjerenja iste mjerene veličine, obavljenih u uslovima ponovljivosti i prave vrijednosti mjerene veličine. Kao i prava vrijednost, tako i sistematska greška i njeni uzroci ne mogu biti u potpunosti poznati. Naime, sistematske greške, po klasičnoj teoriji, potiču od fizičkih veličina koje izazivaju promjenu uvijek iste veličine i znaka, dakle ona je predvidiva. Sistematske greške koje se mogu odrediti eksperimentalno ili proračunom mogu se eliminisati odgovarajućum korekcijom.

Nausuprot tome slučajne greške su prouzrokovane veličinama koje prouzrokuju isključivo slučajne (stohastičke) promjene mjerene veličine. Slučajna greška predstavlja razliku između rezultata mjerenja i srednje vrijednosti koja bi se dobila iz beskonačnog broja mjerenja iste mjerene veličine, obavljenih u uslovima ponovljivosti. Ovakve pretpostavke nisu ispunjene u stvarnim mjerenjima, jer se može obaviti samo konačan broj mjerenja.

Naime često jedna ista fizička veličina izaziva, kako slučajne tako i sistematske efekte. Na primjer, brze fluktuacije temperature imaju za posljedicu efekte koji odgovaraju slučajnim greškama. Međutim, sporije temperaturske promjene mogu stvarati promjene koje odgovaraju sistematskim greškama. Drugi primjer, gdje jedna veličina istovremeno izaziva i brze i spore promjene, je napon napajanja pojačivača i mjernih mostova. Napon napajanja dobijen iz mrežnog ispravljača ima brze fluktuacije (tzv. talasnost), a takođe i lagane promjene koje zavise od temperature okoline i od nestabilnosti mrežnog napona. Otuda napon napajanja istovremeno prouzrokuje i slučajne i sistematske efekte.

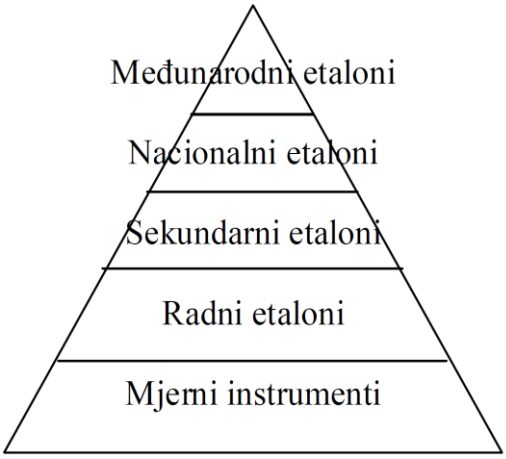
Nasuprot teoriji grešaka, pristup u izražavanju mjerne nesigurnosti dat u *Uputstvu* je praktično orijentisan, pa ga je moguće primijeniti u analizi eksperimentalnih rezultata mjerenja i rezulta etaloniranja.

8.1 ETALONIRANJE I SLIJEDLJIVOST

Etaloniranje instrumenata je potrebno da bi se odredilo i dokumentaovalo odstupanje pokazivanja mjernog instrumenta od stvarne vrijednosti. To znači da instrument treba uporediti sa fizičkim etalonom. Pošto ovo nije uvijek moguće poređenje treba izvršiti po fazama. U svakoj fazi poređenja od etalona do instrumenta poređenje se uvijek vrši između dva instrumenta. Prvo između najvišeg (primarnog) etalona sa nižim (sekundarnim). Zatim između nižeg i sljedećeg nižeg sve dok se poređenje ne završi. Hijararhija laboratorija za etaloniranja prikazana je na slici 8.1. Hijararhija etalona prikazana je na slici 8.2. Ova metoda stvara hijararhiju etalona i instrumenta. Na vrhu je primarni etalon, a

na dnu preko mjernih instrumenata proizvod, koji se proizvodi. Termin slijedljivost ima dva značenja. Prvo, to je osobina rezultata mjerenja ili vrijednosti etalona preko koga se može dovesti u vezu sa nacionalnim ili međunarodnim primarnim etalonom preko neprekidnog lanca poređenja uz utvrđenu mjernu nesigurnost. Drugo, to je mogućnost da se utvrdi istorijat, primjena ili lokacija instrumenata ili aktivnosti putem registrovane identifikacije. Druga definicija se odnosi naprimjer na praćenje traga instrumenta, koji su u laboratoriji etalonirani ili korekcija pogrešnih rezultata etaloniranja. U odnosu na prvu definiciju treba napomenuti da nesigurnost svake karike u lancu mora biti poznata. Ovo se može postići slijedljivim etaloniranjem prema višim etalonima. Tokom svakog poređenja mjerna nesigurnost je sve veća zbog uslova ponovljivosti i uslova okoline. U svakoj fazi poređenja se mjerna nesigurnost povećava i ne može se nikada smanjiti.

Slika 8.1 Etaloniranje



Slika 8.2 Slijedljivost

Važno gledište efikasnosti sistema etaloniranja je određivanje perioda verifikacije etaloniranog mjernog instrumenta.

Period verifikacije zavisi od:

- Parametara za vrijednovanje radnih metoda,
- Tipa instrumenta,
- Preporuke proizvođača,
- Perioda verifikacije sličnih instrumenata,
- Nastajanje kvarova tokom perioda verifikacije,
- Intenziteta korišćenja instrumenta,
- Tendencije trošenja i drifta,
- Uticaja okoline (temperatura, vlažnost, vibracije itd.),
- Podaci o trendu dobijeni iz zapisa predhodnog etaloniranja,
- Zahtjevane nesigurnosti mjerenja,
- Pisanoj istoriji održavanja i servisiranja instrumenata.

Da bi se našao optimalan period verifikacije, treba uspostaviti ravnotežu između dva osnovna, suprotna kriterijuma:

- Rizik da etalon ili mjerni instrument bude izvan zahtjevane tolerancije, treba da je što manji.
- Troškovi etaloniranja treba da su na razumnom nivou.

Prilikom etaloniranja vodi se računa da najveća dopuštena greška etalona bude najmanje 3 puta manja od greške mjerila koje se etalonira. Period verifikacije instrumenata kreće se najčešće od jedne godine za digitalne mjerače vremena i frekvencije, do 5 godina za analogne instrumente.

8.2 IZRAŽAVANJE MJERNE NESIGURNOSTI

Mjerna nesigurnost izražava interval u kome se nalazi prava vrijednost fizičke veličine, koja se mjeri i može se procijeniti ili eksperimentalno odrediti uz određene uslove, koji ograničavaju njenu vrijednost. Jednostavnije rečeno, mjerna nesigurnost je parametar, pridružen rezultatu mjerenja, koji karakteriše rasipanje vrijednosti koje se opravdano mogu pripisati mjerenoj veličini. Usvojeno je da se mjerna nesigurnost označava slovom *u*, (engl. *Uncertainty* - nesigurnost). Osnovni princip je da se svakom podatku o nesigurnosti pridruži odgovarajuća funkcija raspodjele kao i vjerovatnoća, odnosno statistička sigurnost. Osim uobičajenih pojmova iz matematičke statistike, koriste se sljedeći novi pojmovi.

Standardna mjerna nesigurnost u, po definiciji jednaka je standardnom odstupanju od srednje vrijednosti $u = s_{x_s}$. Standardno odstupanje od srednje vrijednosti s_{x_s} računa se prema formuli 8.3. A rezultat više ponovljenih mjerenja predstavlja srednju vrijednost x_s, koja se računa prema formuli 8.1. Statistička sigurnost koja odgovara standardnoj mjernoj nesigurnosti zavisi od raspodjele koja se pripisuje datom mjerenju. Na primjer u slučaju Gausove (normalne) raspodjele mjernih rezultata, intervalu širine jednog standardnog odstupanja, $x_s \pm s$, odgovara sigurnost od 68,2 %, a intervalu širine dva standardna odstupanja, $x_s \pm 2s$, odgovara sigurnost od 95,4 %.

Proširena mjerna nesigurnost, U , predstavlja umnožak standardne mjerne nesigurnosti i *koeficijenta proširenja, k,* tj. $U = k \cdot u$. Koeficijent k može imati vrijednost u intervalu od $\sqrt{3}$ do 3, zavisno od raspodjele. Proširenoj mjernoj nesigurnosti odgovara visoka vrijednost statističke sigurnosti. To znači da se mjerena veličina sa velikom sigurnošću nalazi u intervalu $x_s \pm U$.

U slučajevima kada se rezultatima mjerenja može pripisati Gausova raspodjela i standardna nesigurnost pridružena izlaznoj procjeni je dovoljno pouzdana, treba koristiti standardni koeficijent proširenja $k = 2$. Pripisanoj proširenoj nesigurnosti odgovara vrijednost statističke sigurnosti od približno 95 %. Uslovi za Gausovu

raspodjelu su ispunjeni u većini slučajeva, koji se sreću pri poslovima etaloniranja. Pretpostavka Gausove raspodjele se ne može uvijek eksperimentalno jednostavno potvrditi. Međutim, u slučajevima kada se nekoliko komponenti nesigurnosti (na primjer $N \geq 3$), izvedenih iz vjerovatnoće raspodjele nezavisnih veličina koje se dobro ponašaju (npr. Gausove ili pravougaone raspodjele), doprinose standarnoj nesigurnosti u uporedivim iznosima. Onda se na osnovu centralne granične teoreme može veoma približno pretpostaviti da je raspodjela izlazne veličine Gausova. Za preostale slučajeve tj. sve slučajeve u kojima se ne može prihvatiti pretpostavka Gausove raspodjele, moraju se primijeniti podaci o stvarnoj vjerovatnoći raspodjele radi dobijanja vrijednosti koeficijenta proširenja k, koji odgovara nivou statističke sigurnosti od približno 95 %.

Nakon izvršenog mjerenja akreditovane metrološke laboratorije izdaju sertifikat o etaloniranju ili ispitivanju. U sertifikatima o etaloniranju ili ispitivanju mora se dati kompletan rezultat mjerenja sastavljen od procjene mjerne veličine x_s i pridružene proširene mjerne nesigurnosti U (tj. kao $x_s \pm U$). Uz to treba dati napomenu sa objašnjenjem koja, uglavnom, treba da sadrži koeficijent proširenja k, funkciju raspodjele izmjerenih vrijednosti i vrijednost statističke sigurnosti. Da bi jednoznačno i pouzdano identifikovao rezultate mjerenja sertifikat o etaloniranju potrebno je da sadrži i:

- Naziv, tip, serijski broj i proizvođača mjerila koje se etalonira.
- Mjernu metodu, odnosno određeni propis prema kome je urađeno etaloniranje.
- Navesti izjavu da je rezultat etaloniranja sljedljiv do međunarodnog etalona ili nacionalnog etalona. Mogu se navesti i metrološke karakteristike etalona, koji je korišten i broj njegovog uvjerenja.
- Uslove okoline u toku mjerenja.
- Datum obavljenog etaloniranja.
- Potpis lica koje je izvršilo mjerenje, kao i potpis njegovog rukovodioca.

Brojčana vrijednost mjerne nesigurnosti treba biti data sa najviše dva značajna broja. Brojčanu vrijednost rezultata mjerenja u završnom izvještaju treba zaokružiti na posljednji značajan broj vrijednosti proširene nesigurnosti određene za taj rezultat mjerenja. Međutim, ako se zaokruživanjem smanjuje vrijednost mjerne nesigurnosti za više od 5 %, treba primijeniti zaokruživanje na više.

8.3 IZVORI NASTAJANJA MJERNE NESIGURNOSTI

Nesigurnost rezultata mjerenja odražava nedostatak potpunih saznanja o vrijednosti mjerene veličine. Potpuna saznanja zahtijevaju beskonačnu količinu informacija o izvršenom mjerenju. Fenomeni doprinosa nesigurnosti i činjenica da se rezultat mjerenja ne može okarakterisati kao jedinstvena vrijednost nazivaju se

izvori nesigurnosti. Praktično su mogući mnogi izvori nesigurnosti pri mjerenju koji uključuju:

1) Nepotpuna definicija mjerne veličine,
2) Nesavršena realizacija definicije mjerne veličine,
3) Nereprezentativno uzorkovanje. Uzorak mjerene veličine ne predstavlja definisanu mjerenu veličinu,
4) Nesavršenost u postavci mjerenja, mjerne metode i mjerne procedure,
5) Neadekvatno poznavanje delovanja uslova okoline na mjerenje ili Nesavršeno mjerenje istih,
6) Subjektivna greška metrologa,
7) Nesigurnost mjerne opreme, konačna rezolucija ili prag osjetljivosti mjerila,
8) Netačne vrijednosti mjernih etalona i referentnih materijala,
9) Netačne vrijednosti konstanti ili drugih parametara dobijenih od spoljašnjih izvora i korićenih pri izračunavanju rezultata,
10) Aproksimacije i pretpostavke unijete u proceduru i metodu mjerenja, softver i izračunavanje,
11) Razlike u rezultatima mjerenja pri ponovljenim mjerenjima mjerne veličine uz prividno jednake uslove.

Ovi izvori nisu neophodno nezavisni. Neki od (1) do (10) faktora mogu doprinijeti faktoru (11).

8.4 TIPOVI MJERNE NESIGURNOSTI

Postoje dva osnovna tipa mjerne nesigurnosti: *tip A* i *tip B*. Ova podjela je zasnovana isključivo na osnovu metoda kojima se nesigurnosti određuju. U određenim uslovima, o kojima će biti reči, koristi se i tzv. *kombinovana mjerna nesigurnost*.

8.4.1 Mjerna nesigurnost tip A

Mjerna nesigurnost tip A određuje se isključivo metodom statističke obrade rezultata. Iz ovog slijedi da mjerna nesigurnost tip A postoji samo ako se radi o mjerenju koje je ponovljeno više puta. Postoje dva podtipa procjene ulazne veličine x tipa A i to kao:

- Eksperimentalno standardno odstupanje srednje vrijednosti,
- Kombinovana procjena.

Eksperimentalno standardno odstupanje srednje vrijednosti računa se, kada se radi o ponovljenim mjerenjima i kada se standardno odstupanje srednje vrijednosti

uzima kao standardna nesigurnost tip A. Ako su rezultati ponovljenih mjerenja predstavljeni uzorkom x_1, x_2, ... , x_i, x_n , pomoću (8.1) može se izačunati srednja vrijednost x_s koja, po pravilu predstavlja krajnju vrijednost mjerenja,

$$x_s = \frac{1}{n}\sum_{i=1}^{n} x_i \ . \tag{8.1}$$

Standardna nesigurnost pojedinih elemenata jednaka je standardnom odstupanju pojedinih rezultata što je dato izrazom (8.2),

$$u_A = s = \sqrt{\frac{\sum_{i}^{n}(x_i - x_s)^2}{n-1}} \ . \tag{8.2}$$

Srednja vrijednost x_s koja predstavlja rezultat mjerenja, ima svoje standardno odstupanje s_{x_s} , koje se zove standardno odstupanje srednje vrijednosti. Standardno odstupanje srednje vrijednosti predstavlja standardnu nesigurnost tip A, mjernog rezultata,

$$u_{Ax_s} = s_{x_s} = \frac{u_A}{\sqrt{n}_A} = \sqrt{\frac{\sum_{i}^{n}(x_i - x_s)^2}{n(n-1)}} \tag{8.3}$$

Mjerna nesigurnost tip A se određuje kada se raspolaže sa nizom od *n* međusobno različitih rezultata ponovljenih mjerenja. Ako je broj *n* dovoljno velik, srednja vrijednost i njeno standardno odstupanje ispunjava uslove Centralne granične teoreme što znači da joj se po pravilu pridružuje Gausova raspodjela (nezavisno od raspodjele kojoj pripadaju elementi uzorka). Pri tome raspodjela polaznih rezultata ne mora biti Gausova.

Primjer 8.1 *(Mjerna nesigurnost tip A)*

Sa preciznim digitalnim frekvencmetrom *Agilent* 53131A izvršeno je deset mjerenja frekvencije, signala koji se generiše pomoću generatora funkcija HP3325A (1 MHZ). Prvo je izvršeno deset mjerenja, kada vremenske baze digitalnog frekvencmetra i generatora funkcija nisu bile sinhronizovane. Zatim je izvršeno 10 mjerenja, kada je izvršena sinhronizacija vremenskih baza digitalnog frekvencmetra i generatora funkcija pomoću standarda vremena *GPS* prijemnika *Symmetricom* 58503B. Potrebno je odrediti standardnu mjernu nesigurnost srednje izmjerene vrijednosti frekvencije prije i poslije sinhronizacije sa *GPS* prijemnikom. Dobijeni rezultati mjerenja prikazani su u tabeli 8.1.

Tabela 8.1 Rezultati mjerenja frekvencije

R.br.	Bez sinhronizacije sa *GPS* prijemnikom	Sinhronizacija sa *GPS* prijemnikom
1.	1,000 001 840 MHz	1,000 000 000 01 MHz
2.	1,000 001 895 MHz	1,000 000 000 00 MHz
3.	1,000 001 802 MHz	1,000 000 000 01 MHz
4.	1,000 001 840 MHz	1,000 000 000 00 MHz
5.	1,000 001 729 MHz	1,000 000 000 01 MHz
6.	1,000 001 593 MHz	1,000 000 000 01 MHz
7.	1,000 001 616 MHz	1,000 000 000 00 MHz
8.	1,000 001 669 MHz	1,000 000 000 01 MHz
9.	1,000 001 686 MHz	1,000 000 000 01 MHz
10.	1,000 001 803 MHz	0,999 999 999 99 MHz

Rješenje

Srednja vrijednost frekvencije za deset izvršenih mjerenja prije sinhronizacije je f_s = 1,0000017473 MHz.

Standardno odstupanje frekvencije pojedinih mjerenja je s = 0,103 Hz. Za standardnu mjernu nesigurnost (Tip A) frekvencije pojedinih mjerenja, dobija se

$u = s_{x_s} = s/\sqrt{10}$ = 0,0327 Hz.

Srednja vrijednost frekvencije za deset izvršenih mjerenja poslije sinhronizacije sa *GPS* prijemnikom je

f_s = 1,000000000005 MHz.

Standardno odstupanje frekvencije pojedinih mjerenja je s = 0,000007 Hz. Za standardnu mjernu nesigurnost (Tip A) frekvencije pojedinih mjerenja, dobija se

$u = s_{x_s} = s/\sqrt{10}$ = 0,0000022 Hz.

Na osnovu analize dobijenih rezultata mjerenja frekvencije može su uočiti da se sinhronizacijom vremenske baze digitalnog frekvencmetra i generatora funkcija sa *GPS* prijemnikom značajno smanjuje komponenta mjerne nesigurnosti tip A.

8.4.2 Kombinovana ili objedinjena mjerna nesigurnost tip A

Kao što je rečeno, mjerna nesigurnost tip A dobija se statističkom obradom većeg broja rezultata ponovljenih mjerenja. Međutim, praktične okolnosti često onemogućavaju da se neko mjerenje ponavlja veći broj puta, jer ponavljanje mjerenja značajno poskupljuje proces etaloniranja ili ispitivanja. Ako se mjerenje izvrši samo jedanput ili manji broj puta, na primjer, dva ili tri puta, dobija se uzorak koji je suviše mali da bi se njegovom obradom došlo do valjanih statističkih podataka.

Mjerna nesigurnost, kod ovako malih uzoraka, može se odrediti na osnovu podatka koji se naziva kombinovana[9] ili objedinjena nesigurnost[10]. Pri pregledu nekog instrumenta u metrološkoj laboratoriji vrši se njegovo detaljno ispitivanje. Ono uključuje i višestruko ponavljanje mjerenja pri istim uslovima. Na osnovu jedne ili nekoliko serija ponovljenih mjerenja, određuje se standardno eksperimentalno odstupanje, tj. standardna mjerna nesigurnost tip A.

Kombinovana procjena mjerne nesigurnosti tip A primjenjuje se ukoliko su mjerenja dobro karakterisana i pod statističkom kontrolom. Za mjerenja koja su dobro karakterizirana i statistički se kontrolišu može se koristiti kombinovana ili objedinjena procjena varijanse s_p^2 koja bolje karakteriše rasipanje od procijenjenog standardnog odstupanja postignutog ograničenim brojem mjerenja. Ako imamo slučaj određivanja vrijednosti ulazne veličine X kao aritmetičke sredine x_s malog m broja nezavisnih mjerenja, varijansa srednje vrijednosti može biti procijenjena pomoću jednačine:

$$s^2(\bar{x}) = \frac{s_p^2}{m} \tag{8.4}$$

Kombinovana procjena varijanse s_p^2 zasnovana na N nezavisno ponovljenih serija mjerenja slučajne promjenljive se dobija iz jednačine:

$$s_p^2 = \frac{\sum_{i=1}^{N} v_i s_i^2}{\sum_{i=1}^{N} v_i} \tag{8.5}$$

pri čemu je s_i eksperimentalno standardno odstupanje i-te serije od n_i nezavisnih ponovljenih mjerenja (koje se računa prema formuli 8.2) i ima broj stepeni slobode $v_i = n_i - 1$. Broj stepeni slobode kombinovane varijanse je

$$v = \sum_{i=1}^{N} v_i \ . \tag{8.6}$$

Standardna nesigurnost se izvodi iz ove vrijednosti primjenom jednačine

$$u_{Ap}(x) = s(\mathrm{x_s}) \tag{8.7}$$

---

[9] *D*ragan Stanković, Predrag Osmokraković, *Praktikum laboratorijskih vežbi iz fizike*, Zavod za fiziku tehničkih fakulteta u Beogradu, 2005

[10] Engleski: *Pooled uncertainty*

pa je standardna nesigurnost zasnovna na N nezavisno ponovljenih serija mjerernja, pri čemu serija sa malim brojem uzoraka, za koju se računa mjerna nesigurnost tip A, ima *m* mjerenja:

$$u_{Ap}(x) = \frac{s_p}{\sqrt{m}} \tag{8.8}$$

pri čemu broj stepeni slobode za *u(x)* takođe iznosi *v*.

Primjer 8.2 *(Kombinovana procjena mjerne nesigurnosti tip A)*

Sa preciznim digitalnim frekvencmetrom *Agilent* 53131A izvršeno je pet serija mjerenja u pet različitih dana. Svakoga dana po istoj mjernoj metodi i pri približno istim uslovima okoline rađeno je po deset mjerenja frekvencije signala, koji se generiše pomoću generatora funkcija HP3325A (1 MHZ). Prilikom mjerenja je izvršena sinhronizacija vremenskih baza digitalnog frekvencmetra i generatora funkcija pomoću standarda vremena *GPS* prijemnika *Symmetricom* 58503B. Na ovaj način je smanjena mjerna nesigurnost pojedinačnih mjerenja, kao što je to opisano u predhodnom primjeru. Dobijeni rezultati za seriju mjerenja prikazani su u tabeli 8.1. Za svaku seriju od deset mjerenja izračunata je standardna devijacija s_i prema formuli 8.2. Potrebno je odrediti kombinovanu procjenu mjerne nesigurnosti tip A i broj stepeni slobode, ako se mjerenje frekvencije u jednoj seriji ponavlja 5 ili 10 puta.

Tabela 8.2 Pet serija mjerenja frekvencije

R. br.	Serija 1 [MHz]	Serija 2 [MHz]	Serija 3 [MHz]	Serija 4 [MHz]	Serija 5 [MHz]
1.	1,00000000001	1,00000000001	1,00000000000	1,00000000000	1,00000000001
2.	1,00000000000	1,00000000000	1,00000000000	1,00000000001	1,00000000000
3.	1,00000000001	1,00000000000	1,00000000002	1,00000000000	1,00000000001
4.	1,00000000000	1,00000000003	1,00000000002	1,00000000000	1,00000000000
5.	1,00000000001	1,00000000000	1,00000000000	1,00000000000	1,00000000001
6.	1,00000000001	1,00000000001	1,00000000001	1,00000000001	1,00000000000
7.	1,00000000000	1,00000000001	0,99999999999	1,00000000000	0,99999999998
8.	1,00000000001	1,00000000001	1,00000000000	1,00000000000	1,00000000000
9.	1,00000000001	1,00000000002	1,00000000001	0,99999999999	0,99999999999
10.	0,99999999999	1,00000000001	0,99999999999	1,00000000000	1,00000000000
s_i	$7{,}071 \bullet 10^{-6}$ Hz	$9{,}428 \bullet 10^{-6}$ Hz	$1{,}075 \bullet 10^{-5}$ Hz	$5{,}674 \bullet 10^{-6}$ Hz	$9{,}427 \bullet 10^{-6}$ Hz

Rješenje

Kombinovna procjena varijanse s_p^2, zasnovna na N=5 nezavisno ponovljenih serija od deset mjerenja slučajne promjenljive, dobija se iz jednačine 8.5 i iznosi $7{,}51 \times 10^{-11}$ Hz.

Kombinovana procjena mjerne nesigurnosti tip A koja se računa prema jednačini 8.6 za pet mjerenja frekvencije ($m = 5$) iznosi $u_{Ap} = 0{,}00000387$ Hz, a za

deset mjerenja (m = 10) iznosi u_{Ap} = 0,00000274 Hz. Broj stepeni slobode za jednu seriju od deset mjerenja je 9. Broj stepeni slobode kombinovane procjene mjerne nesigurnosti tip A za pet mjerenja frekvencije je 45.

8.4.3 Mjerna nesigurnost tip B

Mjerna nesigurnost tip B određuje se svim ostalim metodama, izuzev statističke analize. Mjerna nesigurnost tip B može se odrediti i kod pojedinačnog mjerenja, kada mjerna nesigurnost tip A ne postoji. Valjano određivanje mjerne nesigurnosti tip B pretpostavlja upotrebu svih raspoloživih podataka i saznanja o korićenoj mjernoj opremi, o uticaju parametara okruženja na mjerenje, o raznim vrstama smetnji i dr. Pri tome je poželjno da je metrolog koji vrši mjerenje iskusan, da ima što bolja teorijska znanja i da poznaje uticaj okruženja na mjerni proces.

Standardna mjerna nesigurnost, tip B, u_B , predstavlja standardno odstupanje dobijeno analizom različitih uticaja na mjerni rezultat. Pri tome je neophodno da se pridruži adekvatna funkcija raspodjele ovoj nesigurnosti. Kod mjerne nesigurnosti tip B mogu se javiti razne raspodjele (najčešće Gausova, pravougaona i trougaona). Nasuprot tome kod mjerne nesigurnosti tip A uglavnom je uvijek riječ o Gausovoj raspodjeli.

Jedan od najvažnijih izvora podataka za određivanje mjerne nesigurnosti tip B su katalozi i priručnici koje proizvođači daju uz svoj instrument. Obično su dati podaci o nesigurnosti mjerenja u zavisnosti od mjernog opsega, očitanja, rezolucije i pri određenim vrijednostima parametara okoline (opseg temperature, relativna vlažnost i dr).

Prilikom etaloniranja sva mjerila treba da budu pregledana na propisani način pomoću opreme višeg nivoa tačnosti. Podaci u sertifikatu dobijenom nakon etaloniranja instrumenta, zajedno sa svim eventualnim dodatnim objašnjenjima stručnjaka koji su izvršili pregled, predstavljaju korisne izvore u određivanju mjerne nesigurnosti tip B.

Analizom mjerne metode treba ustanoviti, da li dobijeni rezultat zaista predstavlja valjanu brojnu vrijednost fizičke veličine koja treba da se izmjeri. Na primjer, pri mjerenju temperature nekog objekta treba ustanoviti, da li je termički kontakt termometra i objekta dobar, tj. ustanoviti da li termometar zaista mjeri traženu temperaturu.

Pri mjerenju otpornosti nekog otpornika, pored ostalog, treba ustanoviti da li snaga koja se razvija u otporniku izaziva znatan porast temperature, pa time i izmjenu otpornosti. Treba takođe voditi računa o uticaju temperature okoline, vlažnosti i atmosferskog pritiska, ako se radi o mjerenjima visoke tačnosti.

Poznato je da većina laboratorijskih mjerenja ima statički karakter, to jest očitavanje se obavlja tek nakon uspostavljanja stacionarnog stanja. Međutim, potpuna ustaljenost fizičkih veličina nije ostvariva u praksi. Tokom prikupljanja

podataka mjerena veličina i parametri okoline imaju određenu brzinu promjene, pa se javljaju određene dinamičke greške mjerenja. Instrumenti usljed starenja ili primjene u teškim radnim uslovima mogu promjeniti svoje karakteristike. Neophodno je analizirati valjanosti algoritama korićenih u obradi rezultata i dr.

8.4.4 Slučajni karakter sistematskih efekata i nesigurnosti tip B

Prije upotrebe instrumenta neophodno je da svi sistematski efekti, koji utiču na grešku budu potisnuti do minimuma. To se vrši postupkom etaloniranja. Nakon etaloniranja uvijek postoje preostali sistematski efekti čije potiskivanje nije bilo mogućno. Ovi preostali efekti ponašaju se kao slučajne veličine i samim tim imaju svoju funkciju raspodjele. Nesigurnosti tipa B prouzrokovane su sistematskim efektima, koje nije mogućno u potpunosti otkloniti. Odgovarajući primjer je otklanjanje sistematskog efekta kod voltmetra usljed poremećaja "nule", tzv. *nulovanje instrumenta*. Pri nulovanju ulazni krajevi ispitivanog voltmetra se kratko spajaju. Pri tome, umjesto očekivanog nultog napona, na izlazu se očitava napon od, recimo, + 200 μV. Mjerenje sa ovako razdešenim voltmetrom davalo bi rezultate koji su sistematski pomjereni za približno toliki napon. Otuda je neophodno da se prije početka rada otkloni ovaj sistematski poremećaj. Kod instrumenta starije proizvodnje, to se obavlja pomoću odvijača kojim se okreće potenciometar "nula" i pokazivanje indikatora dovodi do nulte vrijednosti. Kod novijih voltmetra to se postiže pritiskom na taster "nula", ili pri automatskom procesu provjere i autokalibracije pri svakom uključenju instrumenta. Međutim, nakon nulovanja mogućno je uočiti da se na indikatoru kratko spojenog instrumenta ipak javljaju manje promjene pokazivanja reda veličine nekoliko μV. To je tzv. kratkotrajna nestabilnost nule. Ako bi se posmatranje produžilo, uočilo bi se dodatno sporije "šetanje" (drift) nule, tako da sistematski pomjeraj na izlazu ipak nije u potpunosti otklonjen. Dakle zaostali sistematski efekati nakon korekcije (etaloniranja) imaju slučajni karakter pa im je s toga potrebno pridružiti neku od odgovarajućih funkcija raspodjele.

Izbor raspodjele za mjerne nesigurnosti tip B može se izvršiti pomoću serije mjerenja. Time se dobija probni uzorak podataka sa kojima se crta histogram i sprovodi ispitivanje o tome koja je raspodjela adekvatna. Za određivanje pogodne raspodjele koriste se određeni statistički testovi. Slučajni karakter mjerne nesigurnosti tip B ilustruje se takođe i sljedećim primjerima.

Primjer 8.3 (*Mjerna nesigurnost usljed rezolucije digitalnog instrumenta*)

Odrediti mjernu nesigurnost idealno tačnog digitalnog instrumenta, uzimajući u obzir samo rezoluciju instrumenta.

Rješenje

Na skali instrumenta očitava se vrijednost mjerene veličine prikazana brojem od N digita. Stvarna vrijednost mjerene veličine, pri tome, ima neku analognu vrijednost koja se sa podjednakom vjerovatnoćom može nalaziti u intervalu između $N - ½$ do $N + ½$ digita. U ovom slučaju sama rezolucija digitalnog instrumenta od jednog digita, unosi proširenu nesigurnost od $a = 0{,}5$ digita. Svaka vrijednost unutar gornjeg intervala podjednako je vjerovatna. Otuda se nesigurnosti usljed rezolucije digitalnih instrumenata pridružuje pravougaona raspodjela, čije je standardno odstupanje $s = \frac{a}{\sqrt{3}}$, pa je standardna nesigurnost

$$u_B = s = \frac{0{,}5}{\sqrt{3}} = 0{,}29 \text{ digita.}$$

Primjer 8.4 (*Mjerna nesigurnost na osnovu podataka proizvođača*)

Komercijalni voltmetar pokazuje na svojoj skali napon $U_V = 1{,}000000$ V. U katalogu proizvođača nalazi se podatak da u tom mjernom opsegu maksimalna nesigurnost iznosi 0,002 % od očitane vrijednosti plus 6 digita. Odrediti proširenu nesigurnost .

Rješenje

Podatak iz kataloga ima smisao proširene nesigurnost U, jer je riječ o garantovanim karakteristikama. Proširena nesigurnost iznosi

$$U = U_V \bullet 0{,}00002 \; + \; 0{,}000006 = 26 \; \mu\text{V}.$$

Iz tog podatka slijedi da se stvarna vrijednost napona nalazi u opsegu od

$$U_V = 1{,}000000 \text{ V} \; \pm \; 26 \; \mu\text{V}.$$

Uslov za izračunavanje standardne nesigurnosti je određivanje odgovarajuće raspodjele na osnovu nekih dodatnih saznanja ili iskustava. Treba primjetiti da kod ovog instrumenta nije neophodno uzimati u obzir nesigurnost usljed digitalnog očitavanja (0,5 digita), jer je znatno manja od proširene nesigurnosti od 26 digita.

8.5 FUNKCIJE RASPODJELE

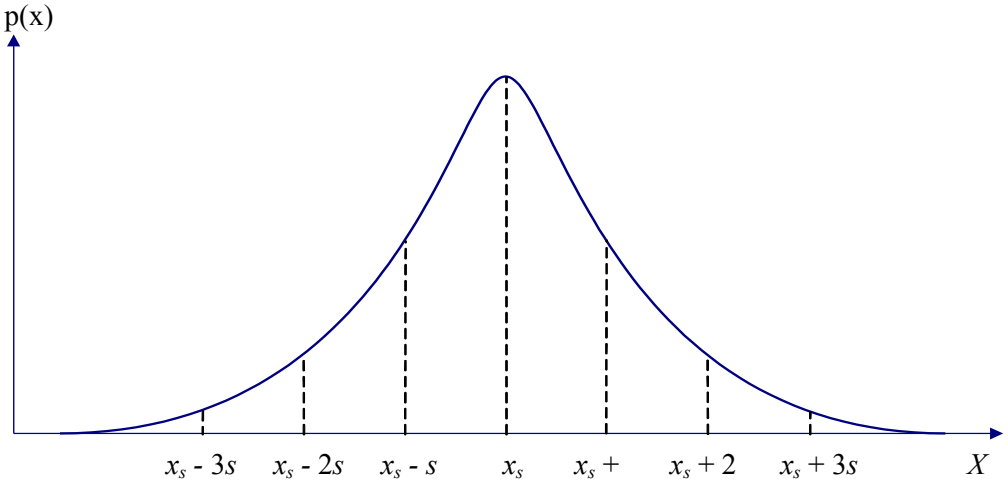
Jedna od osnovnih postavki u konceptu mjerne nesigurnosti je da se svakom sistematskom efektu pridruži neka funkcija raspodjele. Samim tim može se odrediti i vjerovatnoća koja odgovara tom podatku. U pogledu pridruživanja raspodjele postoji jasna razlika koncepta mjerne nesigurnosti u poređenju sa klasičnom teorijom grešaka. U klasičnoj teoriji, funkcija raspodjele se pridružuje samo slučajnim grešakama. Sistematske greške se u toj teoriji posmatraju kao determinističke veličine, tj. kao veličine koje nemaju slučajni karakter nego imaju konstantnu vrijednost u posmatranom mjerenju.

Pri obradi mjernih rezultata primjenjuje se nekoliko vrsta raspodjela. U ovoj knjizi razmotriće se sljedeće raspodjele:

- Gausova (normalna),
- Pravougaona (uniformna) i
- Trougaona.

8.5.1 Gausova (normalna) raspodjela

Eksperimentalno iskustvo pokazuje da se pri ponavljanju nekog mjerenja rezultati na neki način grupišu oko srednje vrijednosti. Na bazi eksperimentalnog iskustva i teorijskih razmatranja Gaus je početkom XIX vijeka izveo raspodjelu kojom se uspješno prikazuju rezultati mjerenja praćeni slučajnim greškama. Ova raspodjela nazvana je Gausovom ili normalnom. Grafički prikaz Gausove raspodjele dat je na slici 8.3. Parametri Gausove raspodjele su *srednja vrijednost* x_s i *standardno odstupanje* s. Oko srednje vrijednosti podjednako vjerovatno su raspoređena pozitivna i negativna odstupanja. U opsegu (srednja vrijednost ± standardno odstupanje) nalazi se oko 68 % rezultata. Ovo se može iskazati i drugim riječima *statističkoj sigurnosti od* 68 % *odgovara interval rezultata* ($x_s - s$, $x_s + s$). Važi i sljedeće pravilo: svaki pojedinačni rezultat ima vjerovatnoću od 68 % da se nađe u intervalu ($x_s - s$, $x_s + s$). Na sličan način se pokazuje da opsegu $x_s \pm 2s$ odgovara statistička sigurnost od 95,4 %. U opsegu $x_s \pm 3s$ nalazi se 99,7 %, tj. u tom intervalu su praktično svi mjerni rezultati. Interval $x_s \pm 3s$ se naziva *maksimalna greška*. Ako je u nekom mjerenju poznato standardno odstupanje s, onda se rezultati sa greškom većom od $\pm 3s$ obično odbacuju jer najvjerovatnije potiču od neke grube greške.

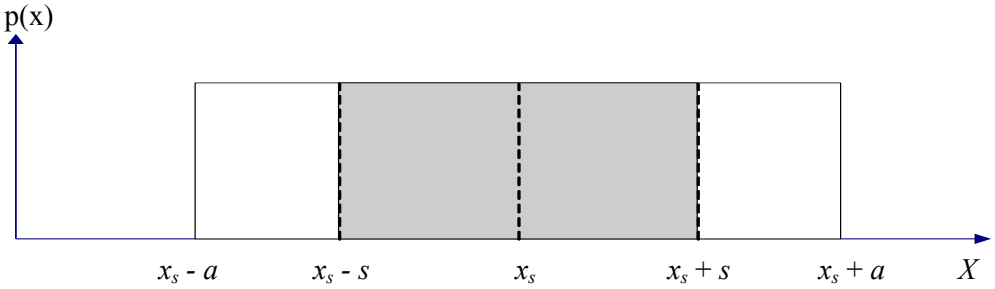


Slika 8.3 Gausova raspodjela

8.5.2 Pravougaona (uniformna) raspodjela

Dijagram simetrične pravougaone raspodjele prikazan je na slici 8.4. Raspodjela je određena srednjom vrijednošću x_s i poluširinom intervala a. Sve vrijednosti slučajne promjenljive x nalaze se u opsegu $x \in (x_s - a, x_s + a)$, pri čemu je svaka vrijednost unutar opsega podjednako vjerovatna. Pravougaona raspodjela ispunjava uslov normiranosti, što znači da površina ispod krive raspodjele p(x) iznosi 1. Standardno odstupanje za pravougaonu raspodjelu računa se kao

$$s = \frac{a}{\sqrt{3}}. \tag{8.7}$$



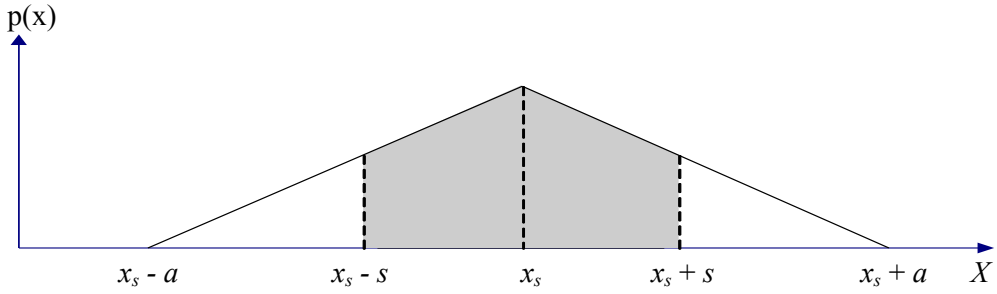
Slika 8.4 Simetrična pravougaona raspodjela sa poluširinom a

U opsegu (srednja vrijednost ± standardno odstupanje) nalazi se oko 57 % rezultata+. Pravougaona raspodjela se najčešće primjenjuje kada se raspolaže sa malo informacija o nekom instrumentu.

Na primjer, iz kataloga proizvođača se pročita podatak da instrument ima klasu tačnosti ±1,5 % maksimalne vrijednosti U_m. Ako ne postoji iskustvo ili drugo saznanje o eventualnom grupisanju rezultata oko srednje vrijednosti, može se pretpostaviti da rezultati pri nekoj vrijednosti mjerene veličine imaju uniformnu raspodjelu sa poluširinom $a = 0{,}015 \cdot U_m$.

8.5.3 Trougaona raspodjela

Dijagram simetrične trougaone funkcije raspodjele prikazana je na slici 8.5. Trougaona i pravougaona raspodjela imaju zajedničku osobinu da im je interval u kojem se nalaze rezultati **ograničen**. Sa slike 8.3. vidi se da se svi rezultati nalaze u ograničenom intervalu poluširine a simetrično raspoređeni oko srednje vrijednosti x_s. Osnovna karakteristika trougaone raspodjele je **skoncentrisanost** rezultata oko srednje vrijednosti. To znači da su manja odstupanja rezultata od srednje vrijednost vjerovatnija od većih odstupanja. Trougaona raspodjela ispunjava uslov normiranosti, što znači da površina ispod krive raspodjele p(x) iznosi 1.



Slika 8.5 Simetrična trougaona raspodjela sa poluširinom a

Standardno odstupanje za trougaonu raspodjelu računa se kao

$$s = \frac{a}{\sqrt{6}}. \tag{8.8}$$

U opsegu (srednja vrijednost mjerenja ± standardno odstupanje) nalazi se oko 65 % svih rezultata. Poredeći ovu dobijenu vrijednost sa statističkom sigurnošću intervala $x_s \pm s$ kod pravougaone raspodjele (57 %) vidi se da trougaona raspodjela, usljed skoncentrisanosti, ima veću sigurnost.

Trougaona raspodjela se primjenjuje u slučaju kada se iz iskustva zna da postoji jasno grupisanje mjernih rezultata oko srednje vrijednosti. Pri tome uslovi centralne granične teoreme nisu potpuno zadovoljeni, što znači da raspodjela najvjerovatnije nije Gausova.

8.5.4 Uticaj izbora raspodjele na izražavanje mjerne nesigurnosti

Radi lakšeg međusobnog poređenja Gausove, pravougaone i trougaone raspodjele u tabeli 8.3. date su statističke sigurnosti i koeficijenti proširenja k, kada se rezultati mjerenja nalaze grupisani oko srednje vrijednosti u nekim intervalima. Iz tabele 8.3. može se lako uočiti da pravougaona raspodjela ima najmanju statističku sigurnost u intervalu $x_s \pm s$ i najmanji koeficijent proširenja.

Tabela 8. 3 Upoređivanje tri funkcije raspodjele

Raspodjela	Interval rezultata mjerenja	Statistička sigurnost	Koeficijent proširenja k
Simetrična Pravougaona	$x_s \pm s$	57,7 %	$\sqrt{3} = 1{,}73$
Simetrična Trougaona	$x_s \pm s$	65 %	$\sqrt{6} = 2{,}45$
Gausova	$x_s \pm s$	68,2 %	1
	$x_s \pm 2s$	95,4 %	2
	$x_s \pm 3s$	99,7 %	3

Primjer 8.5 (*Poređivanje funkcija raspodjele*)

Izvršena je provjera mase 10 proizvoda iste vrste, pri čemu su, pomoću digitalne vage sa rezolucijom 1 g dobijeni podaci dati u tabeli 8.4. Potrebno je odrediti standardnu i proširenu nesigurnost mase pojedinih proizvoda. Ne raspolaže se nikakvim dodatnim podacima u vezi mjerne nesigurnosti.

Tabela 8.4 Mase proizvoda

R.br.	1	2	3	4	5	6	7	8	9	10
Masa [kg]	1,001	0,995	0,997	1,008	0,999	1,011	0,997	1,012	1,008	0,996

Rješenje

Srednja vrijednost mase za deset izvršenih mjerenja je m_s = 1002,4 g. Standardno odstupanje mase pojedinih proizvoda u seriji mjerenja je s = 6,6 g. Za standardnu mjernu nesigurnost (Tip A) mase pojedinih proizvoda ove serije, dobija se $u = s_{x_s} = s/\sqrt{10}$ = 2,1 g. U ovom slučaju je opravdano zanemarivanje standardne nesigurnosti usljed digitalnog očitavanja, koja iznosi $(0{,}5/\sqrt{3} = 0{,}29)$. Iz malog broja dobijenih rezultata nije mogućno pouzdano procijeniti, koja raspodjela je zastupljena.

Proširena nesigurnost U za slučaj tri raspodjele ima sljedeće vrijednosti.

Pravougaona raspodjela: $U = u\sqrt{3} =$ 3,6 g, (P = 57,7 %);

Trougaona raspodjela: $U = u\sqrt{6} =$ 5,1 g, (P = 65 %);

Gausova raspodjela: $U = u \cdot 2$ = 4,2 g, (P = 95,4 %), odnosno
$U = u \cdot 3$ = 6,3 g, (P = 99,7 %).

8.6 KOMBINOVANA MJERNA NESIGURNOST

Kombinovana mjerna nesigurnost je standardna mjerna nesigurnost rezultata mjerenja i koristi se u sljedećim slučajevima:

Za ponovljena mjerenja kod kojih je određena mjerna nesigurnost tip A i istovremeno je za ta mjerenja određena mjerna nesigurnost tip B.

Kada na krajnji rezultat mjerenja utiču nesigurnosti bar dvije ili više uticajnih veličina.

Standardnoj kombinovanoj mjernoj nesigurnosti potrebno je pridružiti adekvatnu funkciju raspodjele. Određivanje standardne kombinovane mjerne nesigurnosti i proširene kombinovane mjerne nesigurnosti po pravilu, predstavlja krajnji cilj obrade mjernih podataka.

8.6.1 Kombinovana mjerna nesigurnost u slučaju nekorelisanih veličina

Dvije veličine su nekorelisane (ili statistički nezavisne) onda, kada promjene jedne od njih, ne izazivaju predvidljive promjene druge veličine. Razmotrimo problem standardnog odstupanja indirektno mjerenih veličina

$$y = f(x_1, x_2, \ldots, x_i, x_n), \tag{8.9}$$

gdje su promjene uticajnih veličina x_i međusobno nezavisne. Zamjenom u izrazu za računanje standardnih odstupanja s_{xi} sa odgovarajućim nesigurnostima, dobija se standardno odstupanje indirektno mjerene veličine

$$u_y = \sqrt{\sum \left[\left(\frac{\partial y}{\partial x_i} \right)^2 \cdot u_{xi}^2 \right]}, \tag{8.10}$$

gdje je u_{xi} standardna mjerna nesigurnost tip B uticajne veličine x_i.

Ako je nelinearnost funkcije $y = f(x_i)$ značajna za izračunavanje kombinovanog standardnog odstupanja koristi se sljedeća jednačina:

$$u_{yc}^2 = \sum_{i=1}^{N} \left(\frac{\partial y}{\partial x_i} \right)^2 \cdot u_{xi}^2 + \sum_{i=1}^{N} \sum_{j=1}^{N} \left(\frac{1}{2} \left(\frac{\partial^2 y}{\partial x_i \partial x_j} \right)^2 + \frac{\partial y}{\partial x_i} \frac{\partial^3 y}{\partial x_i \partial x_j^2} \right) \cdot u_{xi}^2 \cdot u_{xj}^2 \tag{8.11}$$

Parcijalni izvodi $\partial y / \partial x_i$ se nazivaju koeficijenti osjetljivosti. Oni opisuju kako procjena izlazne veličine varira sa promjenom vrijednosti procjene ulaznih veličina $x_1, x_2, x_3, \ldots, x_N$.

Kombinovna standardna nesigurnost procjene izlazne veličine y se izračunava jednačinom:

$$u_{yc} = \sqrt{\sum_{i=1}^{N} \left[c_i u_{x_i} \right]^2} \tag{8.12}$$

gdje je:

$$c_i \equiv \frac{\partial y}{\partial x_i} \tag{8.13}$$

Mjerne nesigurnosti tip A i tip B kod neke serije ponovljenih mjerenja dobijaju se potpuno različitim metodama. To znači da su ove dvije vrste nesigurnosti uvijek nekorelisane slučajne veličine**.** Kada se odrede standardne nesigurnosi u_A i u_B, kombinovana standardna nesigurnost dobija se "sabiranjem" dviju nesigurnosti. Kod zbira više veličina

$$y = x_1 + x_2 + \ldots + x_i + \ldots + x_n = \sum x_i \qquad \text{važi} \quad \frac{\partial y}{\partial x} = 1$$

i (8.12) postaje

$$u_{\mathrm{y}} = \sqrt{\sum u_{xi}^2} \quad . \tag{8.14}$$

Kombinovana mjerna nesigurnost u slučaju dvije nekorelisane nesigurnosti u_{A} i u_{B} iznosi

$$u_{\mathrm{c}} = \sqrt{u_A^2 + u_B^2} \quad . \tag{8.15}$$

***Primjer* 8.6** (*Kombinovana mjerna nesigurnost na osnovu mjerne nesigurnosti tip A i mjerne nesigurnosti tip B*).

Primjenom voltmetra visoke rezolucije, u ustaljenom režimu nekog električnog kola više puta je ponovljeno mjerenje razlike potencijala. Na osnovu dobijenog uzorka razultata određena je srednja vrijednost Vs = 1,003210 V i standardno odstupanje srednje vrijednosti s_{Vs} = 6 μV. U katalogu proizvođača daje se podatak da instrument na tom mjernom opsegu ima maksimalnu nesigurnost ΔV_{m} = 28 μV. Iskustvo u radu sa ovim instrumentom pokazuje da postoji određeno grupisanje rezultata oko srednje vrijednosti. Odrediti standardnu i proširenu kombinovanu mjernu nesigurnost u slučaju date serije ponovljenih mjerenja.

Rješenje

Standardna mjerna nesigurnost tip A srednje vrijednosti mjerenja voltmetra jednaka je standardnom odstupanju $u_A = s_{Vs}$ = 6 μV. Standardna mjerna nesigurnost tip B dobija se iz podatka, koje proizvođač daje o maksimalnoj nesigurnosti uz usvajanje odgovarajuće funkcije raspodjele. Iz uslova zadatka da se rezultati mjerenja grupišu oko srednje vrijednosti slijedi da se nesigurnosti tipa B može osnovano pridružiti trougaona raspodjela sa poluširinom a = 28 μV. (Mogla bi se takođe usvojiti i Gausova raspodjela). Standardna mjerna nesigurnost tip B za trougaonu raspodjelu iznosi $u_B = a/\sqrt{6}$ = 11,43 μV.

U ovom primjeru opravdano je zanemarivanje mjerne nesigurnost usljed digitalnog očitavanja rezultata jer odgovarajuća proširena nesigurnost zadnjeg digita iznosi 0,5 μV, a standardna nesigurnost 0,29 μV, vidjeti primjer (8.3).

Nesigurnosti tip A i tip B uvijek predstavljaju nekorelisane veličine, jer se određuju potpuno različitim postupcima koji su nezavisni. Otuda kada postoje nesigurnosti tip A i tip B standardna kombinovana mjerna nesigurnost određuje se izrazom (8.15),

$$u_{\mathrm{cAB}} = \sqrt{u_A^2 + u_B^2} = 13 \text{ μV} .$$

Proširena kombinovana mjerna nesigurnost dobija se množeći u_{c} koeficijentom proširenja k čija vrijednost zavisi od funkcije raspodjele.

U cilju izbora optimalne funkcije raspodjele kombinovane nesigurnosti razmatraju se sljedeće činjenice.

- Nesigurnosti tipa A, shodno centralnoj graničnoj teoremi odgovara Gausova raspodjela.
- Nesigurnosti tipa B pridružena je trougaona, dakle jedna takođe skoncentrisana raspodjela.
- Otuda se zaključuje da se kombinovanoj mjernoj nesigurnosti u_{cAB} može opravdano pridružiti neka od skoncentrisanih raspodjela (Gausova ili trougaona).

Koeficijent proširenja za trougaonu raspodjelu iznosi $k = 2{,}45$, a za Gausovu raspodjelu iznosi $k = 2$ (P = 95 %), (vidi tabelu 8.3.). Kao kompromisna vrijednost, može se usvojiti $k = 2{,}2$, što je približno srednja vrijednost gornjih koeficijenata. Dakle, proširena kombinovana mjerna nesigurnost, kojoj odgovara statistička sigurnost bliska 95 %, iznosi $U_c = k \cdot u_c \approx 28{,}6\ \mu\text{V}$.

Primjer 8.7 *(Kombinovana mjerna nesigurnost na osnovu nekorelisanih mjernih nesigurnosti tipa B*).

U cilju dobijanja otpornika nominalne otpornosti $R_e = 1000\ \Omega$ ostvarena je redna veza deset otpornika nominalne otpornosti $R = 100\ \Omega$. Proširena nesigurnost (maksimalno odstupanje) svakog od otpornika iznosi $U_R = 0{,}1\ \Omega$. Otpornici i podaci o njima potiču od različitih izvora (proizvođača ili laboratorija). Odrediti proširenu nesigurnost ekvivalentne otpornosti U_{Re}.

Rješenje

Proširenoj mjernoj nesigurnosti pojedinih otpornika, koji se nalaze u rednoj vezi, pridružuje se pravougaona raspodjela, pa se za standardnu mjernu nesigurnost jednog otpornika dobija

$$u_R = U_R / \sqrt{3} = 0{,}058\ \Omega.$$

Ekvivalentna otpornost iznosi $R_e = \sum R_i$, odakle slijedi $\dfrac{\partial \mathrm{Re}}{\partial \mathrm{R_i}} = 1$. Prema uslovu zadatka, pojedine otpornosti R_i etalonirane su različitom opremom, pa se mogu smatrati nekorelisanim veličinama. Primjenom izraza (8.15) dobija se

$$u_{Re} = u_R \cdot \sqrt{10} = 0{,}18\ \Omega.$$

Ekvivalentna otpornost R_e, kao zbir većeg broja slučajnih veličina, ispunjava uslove centralne granične teoreme, pa ima približno Gausovu raspodjelu. Usvajajući koeficijent proširenja $k = 2$ (P = 95 %), proširena mjerna nesigurnost ekvivalentne otpornosti iznosi

$$U_{Re} = k \cdot u_{Re} = 0{,}36\ \Omega,$$

pa je

$$R_e = (1000 \pm 0{,}36)\ \Omega.$$

8.6.2 Kombinovana mjerna nesigurnost u slučaju korelisanih veličina

U klasičnoj teoriji grešaka razmatraju se sistematske greške indirektno mjerenih veličina koje se mogu izraziti u obliku y = f(x_1, x_2, ... , x_i, x_n). Kada su promjene uticajnih veličine x_i u potpunosti korelisane sve promjene se sabiraju. Ako se promjene Δy i Δx_i Zamijene svojim standardnim nesigurnostima u_y i u_{xi} , respektivno, dobija se

$$u_y = \sqrt{\sum_{i=1}^{N} c_i^2 u^2{}_{xi}) + 2\sum_{i=1}^{N}\sum_{j=i+1}^{N} c_i c_j u_{xj} r(x_i, x_j)} \,. \tag{8.16}$$

pri čemu je:

$$r(x_i, x_j) = \frac{u(x_i, x_j)}{u_{xi} \cdot u_{xj}} \tag{5.17}$$

i to predstavlja formulu za korelacione koeficijente (pri čemu je $i \neq j$ i $| r | \leq 1$). Ako su srednje vrijednosti korelisanih ulaznih veličina x_i i x_j izračunate na osnovu n nezavisnih parova simultanih mjerenja pod istim uslovima tada je:

$$u(x_i, x_j) = s(\bar{x}_i, \bar{x}_j)$$

pri čemu je:

$$s(\bar{x}_i, \bar{x}_j) = \frac{1}{n(n-1)} \sum_{k=1}^{n} (x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j) \tag{5.18}$$

U specijalnom slučaju kada je $\left|\frac{\partial y}{\partial x_i}\right| = 1$ (funkcija y predstavlja zbir ili razliku veličina x_i i tada su korelacioni koeficijenti jednaki jedan), nesigurnost je data zbirom nesigurnosti mjerenih veličina

$$u_y = \sum u_{xi} \tag{8.19}$$

Treba imati u vidu da nesigurnosti i greške (slučajne ili sistematske) predstavljaju različite veličine. Dok odstupanja izmjerene od srednje vrijednosti mogu biti različitog znaka, dotle nesigurnost u_y kao i nesigurnosti uticajnih veličina uxi uvijek su pozitivne veličine.

Primjer 8.8 (K*ombinovana mjerna nesigurnost na osnovu korelisanih mjernih nesigurnosti tip B*).

Otpornik nominalne otpornosti R_e = 1000 Ω dobijen je rednom vezom deset istih otpornika otpornosti R = 100 Ω. Proširena mjerna nesigurnost (maksimalno odstupanje) svake od otpornosti iznosi U_R = 0,1 Ω. Sve otpornosti su određene

istom opremom i pomoću istog etalonskog otpornika. Odrediti nesigurnost ekvivalentne otpornosti U_{Re}.

Rješenje

Ekvivalentna otpornost redne veze otpornika je $R_{\mathrm{e}} = \sum R_i$. Sve otpornosti određene su istovjetnom metodom i opremom pa se mogu smatrati potpuno korelisanim. Primjenom formule (8.19) dobija se $u_{Re} = 10u_R$. U ovom slučaju ekvivalentna otpornost R_{e}, kao zbir korelisanih vrijednosti, ne ispunjava uslove centralne granične teoreme. Otuda je raspodjela R_{e} pravougaona, kao i kod pojedinih otpornosti. Proširena mjerna nesigurnost iznosi

$$U_{\mathrm{Re}} = 10 \text{ x } U_{\mathrm{R}} = 1\ \Omega,$$

pa je

$$R_{\mathrm{e}} = (1000 \pm 1)\ \Omega.$$

Treba obratiti pažnju da u ovom slučaju postoji uočljiva razlika dobijene proširene mjerne nesigurnosti, u odnosu na proširenu mjerna nesigurnost u primjeru 8.7. Mjerna nesigurnost mjerenja sa korelisanim veličinama je veća od mjerne nesigurnosti sa nekorelisanim uticajnim veličinama.

8.7 POSTUPAK IZRAČUNAVANJA MJERNE NESIGURNOSTI

Definisanje mjerne metode i postupka izračunavanja mjerne nesigurnosti u metrološkoj laboratoriji (ispitnoj laboratoriji) potrebno je da se vrši u pisanoj formi obično u sljedećim koracima:

1) Identifikacija mjerne metode i postupka,
2) Definisanje jednačine modela,
3) Kvantifikovanje komponenti nesigurnosti,
4) Određivanje kombinovane mjerne nesigurnosti,
5) Određivanje proširene mjerne nesigurnosti,
6) Analiza podataka i rezultatata.

8.7.1 Identifikacija mjerne metode

Mjerna metoda i postupak izračunavanja mjerne nesigurnosti za svaku vrstu etaloniranja (ispitivanja) koja se sprovodi u metrološkoj laboratoriji (ispitnoj laboratoriji) mora biti jednoznačno identifikovana i označena. Mjerna metoda mora imati naziv, kratak opis i standard prema kom se primjenjuje mjerna metoda. Jednoznačna identifikacija se sastoji u tome da se jednoj mjernoj metodi i odgovarajućem postupku izračunavanja mjerne nesigurnosti pridruži jedinstven

identifikacioni broj. U mjernoj metodi se navode svi etaloni (koji takođe trebaju biti jednoznačno identifikovani) sa kojima se vrši mjerenje, kao i karakteristike mjerila koje se etaloniraju, što dodatno jednoznačno određuje mjernu metodu i postupak. Za svaki etalon treba biti navedena mjerna nesigurnost sa kojom je on etaloniran i kako se ostvaruje sljedljivost do međunarodnog etalona.

8.6.2 Definisanje jednačine modela

Procjena mjerene veličine Y, se označava sa y a dobija se iz jednačine (8.20) koristeći procjene ulaznih veličina X_i koje se označavaju sa x_i:

$$y = f(x_1, x_2, x_3, \ldots x_N) \tag{8.20}$$

Funkcionalna zavisnost f koja odražava princip, metodu i postupak mjerenja se definiše kao model mjerenja, koji opisuje kako se dobija procjena vrijednosti mjerene veličine y iz procjene vrijednosti ulaznih veličina x_i. Procjene ulaznih veličina $x_1, x_2, x_3, \ldots x_N$ od kojih zavisi procjena izlazne veličine y mogu, same po sebi, biti zavisne od procjene drugih veličina, uključujući korekcije i korekcione faktore za sistematske efekte.

Funkcionalna zavisnost f može biti:

- eksplicitna,
- sastavljena od više funkcija,
- eksperimentalno određena,
- algoritam,
- kombinacija gore navednih oblika.

Model funkcije f se mora izražavati tako da se u nju uključe sve one veličine, koje u značajnoj mjeri mogu doprinijeti mjernoj nesigurnosti.

Ako analiza podataka ukazuje na to da funkcija f nije adekvatna stepenu zahtijevane tačnosti mjernih rezultata, u funkciju f se moraju uključiti dodatne ulazne veličine da bi se eliminisala ta neadekvatnost.

Ako ulazne vrijednosti nisu korigovane zbog uticaja bitnih za model, neophodne korekcije treba uvesti kao posebnu ulaznu veličinu.

Model funkcije f treba pisati u takvoj formi da se broj korelisanih ulaznih veličina svede na najmanju moguću mjeru.

Standardna mjerna nesigurnost pridružena procjeni y mjerene veličine Y, koja se označava sa $u(y)$, je standardno odstupanje mjerene veličine Y. Ona se određuje iz procjene x_i ulaznih veličina X_i i njima pripadajuće standardne nesigurnosti $u(x_i)$. Standardna nesigurnost sa pridruženom procjenom ima istu dimenziju kao i procjena. U nekim slučajevima se može koristi i relativna standardna mjerna nesigurnost mjerenja dobijena kada se mjerna nesigurnost procjene podjeli sa apsolutnom vrijednošću procjene te je prema tome bez dimenzije. Ovaj koncept se ne može koristiti, ako je procijenjena vrijednost jednaka nuli.

Prilikom pisanja jednačina modela mjerenja potrebno je pridržavati se određenih pravila. Model mjerenja može biti dat sa jednom ili više jednačina pri čemu je obavezno za svaku veličinu u jednačini definisati oznaku, jedinicu i naziv. Ulazne veličine koje se unose u model su podijeljene na glavne, nominalne, korekcione i veličine granica pri čemu su za svaku od ovih ulaznih veličina definisana sljedeća pravila pisanja:

a) Glavne veličine su one ulazne veličine koje bitno doprinose mjerenoj veličini. One se pišu malim slovima italikom. Ukoliko predstavljaju razliku ispred njih se dodaje veliko grčko slovo delta.

Primjer 8.9

t_{ix} – temperatura pokazivanja termometra X, koji se etalonira. Indeks *i* znači pokazana (indikovana) vrijednost.

Δl – razlika u pomijeraju prilikom mjerenja dužine.

b) Nominalne veličine su veličine pridružene realizaciji veličine preko etalona ili mjernog instrumenta. One su približne veličine koje čine glavni dio realizovane veličine. One se pišu velikim slovima italikom.

Primjer 8.10

L – nominalna dužina paralelne granične mjerke koja se etalonira.

c) Korekcione veličine su mala odstupanja od glavne vrijednosti, koja su poznata ili koja se moraju odrediti. U većini slučajeva korekcione veličine su aditivne. Pišu se slovom koje predstavlja veličinu koju razmatramo ispred kojeg se dodaje malo grčko slovo delta (δ).

Primjer 8.11

δm_p – moguće odstupanje vrijednosti etalonskog tega od njegovog zadnjeg etaloniranja.

δm_c – korekcija zbog ekscentriteta opterećenja i magnetnih efekata pri etaloniranju tega.

d) Veličine granica su utvrđene (fiksne) vrijednosti mogućih varijacija vrijednosti veličina. One se pišu slovom (simbolom) kojim je označena veličina koja se razmatra ispred kojeg se dodaje veliko grčko slovo delta.

Primjer 8.12.

$\Delta\alpha_x$ – procijenjena poluširina intervala mogućeg odstupanja linearnog termičkog koeficijenta otpornosti otpornika, koji je dat u specifikaciji proizvođača za otpornik, koji se etalonira.

e) Definisane referentne vrijednosti pišu se simbolom koji predstavlja veličinu sa indeksom 0.

Primjer 8.13.

p_0 **-** referentni pritisak, npr. 1000 mbar.

f) Odnosi veličina iste vrste (bezdimenzionalni odnosi) se pišu malim slovima italikom.

Primjer 8.14.

$r = R_x / R_{iN}$ **-** odnos pokazivanja otpronika čiji je otpor nepoznat i etalonskog otpornika.

Ako se etaloniranje vrši komparacijom etalona i mjerila koje se etalonira direktno bez posrednika (komparater mjerila), rezultat komparacije je razlika u njihovim pokazivanjima, pa bi matematičku formulu u prvom koraku trebalo pisati u vidu:

d=(pokazivanje etalona x *korekcija etalona)-(pokazivanje mjerila* x *korekcija mjerila)* (8.21)

Procjenu mjerene veličine vršimo mjerilom i označavamo sa *y,* pa je jednačina modela data sa :

y=((pokazivanje etalona x korekcija etalona)-d)/(korekcija mjerila) (8.22)

Ako se etaloniranje vrši komparacijom etalona i mjerila koje se etalonira indirektno preko posrednika (komparater mjerila), rezultat komparacije je korigovana razlika u njihovim pokazivanjima, pa bi matematičku formulu u prvom koraku trebalo pisati:

d x *korekcija komparatera =(pokazivanje etalona* x *korekcija etalona)-(pokazivanje mjerila* x *korekcija mjerila)* (8.23)

Procjenu mjerene veličine vršimo mjerilom i označavamo sa *y,* pa je jednačina modela data sa :

y=((pokazivanje etalona x korekcija etalona)-(d x *korekcija komparatera))/ (korekcija mjerila)* (8.24)

Jednačina modela u oba gore navedena slučaja dobija oblik:

$$y = x_s + d + \delta x + \delta d \cdots \tag{8.25}$$

tako da je u opštem slučaju:

$$y = x_1 + x_2 + \cdots + x_N \tag{8.26}$$

odnosno dobijamo jednačinu modela (8.20)

$$y = f(x_1, x_2, x_3, \ldots x_N) \tag{8.20}$$

8.7.3 Kvantifikovanje komponenti nesigurnosti

Za svaku veličinu procjene x_i u jednačini modela (8.20) daje se opis, koji bliže određuje način na koji se dobijaju podaci o mjernoj nesigurnosti procjene te veličine. Svakoj veličini iz modela bilo da je ona ulazna ili izlazna, obavezno treba dodati oznaku tipa koja određuje prirodu i položaj te veličine u modelu mjerenja kao i tip procjene mjerne nesigurnosti.

Mjerna nesigurnost tip A određuje se isključivo metodom statističke obrade rezultata. Iz ovog slijedi da mjerna nesigurnost tip A postoji, samo ako se radi o mjerenju koje je ponovljeno više puta. Način određivanja mjerne nesigurnosti tip A opisan je u poglavlju 8.4.1.

Mjerna nesigurnost tip B određuje se svim ostalim metodama, izuzev statističke analize. Mjerna nesigurnost tip B može se odrediti i kod pojedinačnog mjerenja, kada mjerna nesigurnost tip A ne postoji. Valjano određivanje mjerne nesigurnosti tip B pretpostavlja upotrebu svih raspoloživih podataka i saznanja o korićenoj mjernoj opremi, o uticaju parametara okruženja na mjerenje, o raznim vrstama smetnji i dr. Pri tome je poželjno da je metrolog koji vrši mjerenje iskusan, da ima što bolja teorijska znanja i da poznaje uticaj okruženja na mjerni proces. Način određivanja mjerne nesigurnosti tip B opisan je u poglavlju 8.4.2.

8.7.4 Određivanje proširene mjerne nesigurnosti

Određivanje kombinovane mjerne nesigurnosti i proširene kombinovane mjerne nesigurnosti po pravilu, predstavlja krajnji cilj obrade mjernih podataka. Kombinovana mjerna nesigurnost je standardna mjerna nesigurnost rezultata mjerenja i koristi se u sljedećim slučajevima:

- Za ponovljena mjerenja kod kojih je određena mjerna nesigurnost tip A i istovremeno je za ta mjerenja određena mjerna nesigurnost tip B.
- Kada na krajnji rezultat mjerenja utiču nesigurnosti bar dvije ili više uticajnih veličina.

Standardnoj kombinovanoj mjernoj nesigurnosti potrebno je pridružiti adekvatnu funkciju raspodjele. Način određivanja kombinovane mjerne nesigurnosti opisan je u poglavlju 8.6.

Svakom rezultatu mjerenja treba da se pridruži odgovarajuća proširena mjerna nesigurnost. *Proširena mjerna nesigurnost, U,* predstavlja umnožak standardne mjerne nesigurnosti i *koeficijenta proširenja, k,* tj. $U = k \cdot u$. Proširenoj mjernoj nesigurnosti odgovara visoka vrijednost statističke sigurnosti. To znači da se mjerena veličina sa velikom sigurnošću nalazi u intervalu $x_s \pm U$.

U slučajevima kada se rezultatima mjerenja može pripisati Gausova (normalna) raspodjela i standardna nesigurnost pridružena izlaznoj procjeni je dovoljno pouzdana, treba koristiti standardni koeficijent proširenja $k = 2$. Pridruženoj

proširenoj mjernoj nesigurnosti odgovara vrijednost statističke sigurnosti od približno 95 %. Uslovi za Gausovu raspodjelu su ispunjeni u većini slučajeva, koji se sreću pri poslovima etaloniranja.

8.7.5 Određivanje faktora pokrivanja na osnovu broja stepeni slobode

Za određivanje vrijednosti faktora pokrivanja k koji odgovara specificiranoj vjerovatnoći pokrivanja potrebno je da se u proračun uzme pouzdanost standardne nesigurnosti $u(y)$ izlazne procjene y. To znači razmotriti kako dobro $u(y)$ procijenjuje standardno odstupanje pridruženo rezultatu mjerenja. Za ocjenu standardnog odstupanja normalne raspodjele, pogodna mjera pouzdanosti je stepen slobode te procjene, koji zavisi od veličine uzorka. Slično pogodna mjera pouzdanosti standardne nesigurnosti, pridružene izlaznoj procjeni, je njen efektivni stepen slobode V_{eff} . Efektivni stepen slobode V_{eff} je približan pogodnoj kombinaciji efektivnog stepena slobode njegovih raznih doprinosa mjerne nesigurnosti $u_i(y)$.

Postupak izračunavanja pogodnog faktora pokrivanja k, kada su zadovoljeni uslovi centralne granične teoreme, obuhvata sljedeća tri koraka:

a) Dobijanje standardne nesigurnosti pridružene izlaznoj procjeni postupkom korak po korak.
b) Procjena efektivnog stepena slobode V_{eff} za kombinovanu standardnu nesigurnost, koji se pridružuje izlaznoj procjeni računa se prema sljedećoj formuli

$$V_{eff} = \frac{u_c^4(y)}{\sum_{i=1}^{N} \frac{u_i^4(y)}{V_i}} \tag{8.27}$$

gdje su:

- $u_c(y)$ kombinovana standardna nesigurnost.
- $u_i(y)$ (i = 1, 2, 3 ... N) doprinosi standarne nesigurnosti pridruženi izlaznoj procjeni y, koja je rezultat standardne nesigurnosti pridružene ulaznoj procjeni x_i, za koje je pretpostavljeno da su međusobno statistički nezavisne.
- V_i je efektivni stepen slobode doprinosa standardne nesigurnosti $u_i(y)$.

Za standardnu mjernu nesigurnost dobijenu procjenom tip A, broj stepeni slobode je dat sa

$$V_i = n\text{-}1.$$

Veći problem je pridružiti stepen slobode standardnoj mjernoj nesigurnosti, koja je dobijena na osnovu procjene mjerne nesigurnosti tip B. Međutim, uobičajena je praksa izvođenje procjena na način koji osigurava izbjegavanje bilo kakvog nižeg ocjenjivanja. Ako se, na primjer,

uvode donje i gornje granice a_- i a_+ , one se obično biraju tako da je izuzetno mala vjerovatnoća da konkretni rezultat mjerenja leži izvan tih granica. Pod pretpostavkom usvajanja takve prakse za stepen slobode standardne nesigurnosti $u(x_i)$ dobijene procjenom tip B može se usvojiti da je $V_i \to \infty$.

c) Usvojiti faktor pokrivanja k na osnovu podataka iz tabele 8.5. Ova tabela je zasnovana na t - raspodjeli za različite vrijednosti vjerovatnoće pokrivanja. Ako V_{eff} nije cijeli broj, što je većinom slučaj, tada V_{eff} zaokružiti na sljedeći manji broj.

Tabela 8.5 Faktor pokrivanja k za razne efektivne stepene slobode V_{eff}

Broj stepeni slobode V_{eff}	Vrijednost vjerovatnoće pokrivanja u procentima					
	68,27%	90%	95%	95,45%	99%	99,73%
1	1,84	6,31	12,71	13,97	63,66	235,80
2	1,32	2,92	4,30	4,53	9,92	19,21
3	1,20	2,35	3,18	3,31	5,84	9,22
4	1,14	2,13	2,78	2,87	4,60	6,62
5	1,11	2,02	2,57	2,65	4,03	5,51
6	1,09	1,94	2,45	2,52	3,71	4,90
7	1,08	1,89	2,36	2,43	3,50	4,53
8	1,07	1,86	2,31	2,37	3,36	4,28
9	1,06	1,83	2,26	2,32	3,25	4,09
10	1,05	1,81	2,23	2,28	3,17	3,96
11	1,05	1,80	2,20	2,25	3,11	3,85
12	1,04	1,78	2,18	2,23	3,05	3,76
13	1,04	1,77	2,16	2,21	3,01	3,69
14	1,04	1,76	2,14	2,20	2,98	3,64
15	1,03	1,75	2,13	2,18	2,95	3,59
16	1,03	1,75	2,12	2,17	2,92	3,54
17	1,03	1,74	2,11	2,16	2,90	3,51
18	1,03	1,73	2,10	2,15	2,88	3,48
19	1,03	1,73	2,09	2,14	2,86	3,45
20	1,03	1,72	2,09	2,13	2,85	3,42
25	1,02	1,71	2,06	2,11	2,79	3,33
30	1,02	1,70	2,04	2,09	2,75	3,27
35	1,01	1,70	2,03	2,07	2,72	3,23
40	1,01	1,68	2,02	2,06	2,70	3,20
45	1,01	1,68	2,01	2,06	2,69	3,18
50	1,01	1,68	2,01	2,05	2,68	3,16
100	1,005	1,660	1,984	2,025	2,626	3,077
∞	1,000	1,645	1,960	2,000	2,576	3,000

8.8 PRIMJER ETALONIRANJA TERMOPARA TIP K

U ovom poglavlju opisana je metoda etaloniranja termopara K tip na 700 ºC, u vertikalnoj kalibracionoj peći, metodom poređena temperature izmjerene sa etalonskim termoparom S tip. Ova metoda[11] napravljena je u Metrološkoj laboratoriji ML-16 u VZ "ORAO". Izrađena je za potrebe akreditacije ove laboratorije kod Bosanskog akreditacionog tijela (BATA). Rezultat etaloniranja je izlazna elektromotorna sila termopara, za temperaturu njegovog toplog spoja, kada se hladni spoj nalazi na referentnoj temperaturi 0 ºC. Ova metoda etaloniranja sastoji se od dva koraka:

- određivanje temperature u peći pomoću etalonskog termopara S tip i
- određivanje elektromotorne sile termopara, koji se etalonira.

Zato je i procjena mjerne nesigurnosti podijeljena u dva dijela i postoje dva matematička modela koja opisuju ovu procjenu. Međutim, ova dva modela su međusobno povezana.

8.8.1 Opis metode

Prilikom etaloniranja termopara tip K na 700 ºC, kao temperaturni izvor koristi se vertikalna kalibraciona peć (tip 912C, proizvođač *Rosemount*). Termopar koji se etalonira postavlja se u termo blok peći pored referentnog termopara tip S (tip 163A, proizvođač *Rosemount*), koji se koristi kao etalon za mjerenje temperature u peći. Potrebno je da vrh termopara koji se etalonira i vrh referentnog termopara budu na istoj visini. Kada peć dostigne temperaturu od 700 ºC, potrebno je sačekati pola sata da se temperatura u peći stabiliše.

Elektromotorna sila koja se stvara na krajevima termopara koji se etalonira, kompenzacionim kablom tip K vodi se do termos boce u kojoj se ostvaruje referentne temperature hladnog spoja 0 ºC. Na referentnoj temperaturi hladnog spoja vrši se spajanje kompenzacionog kabla tip K i bakarnih kablova sa kojim se signal dalje vodi do digitalnog multimetra (tip HP3458A, proizvođač *Hewlett Packard*). Spoj etalonskog termopara tip S i bakarnih kablova sa kojima se signal vodi na digitalni multimetar, vrši se takođe na referentnoj temperaturi hladnog spoja 0 ºC. Spoj kompenzacionog i bakarnog kabla postavlja se u staklenu epruvetu, kako u vodi ne bi došlo do kontakta između dva kraja termopara. Referentna temperatura hladnog spoja od 0 ºC ostvaruje se pomoću destilovane vode i leda koji se miješaju u termos boci. Referentna temperatura hladnog spoja kontroliše se pomoću živinog termometra podjele 0,1 ºC .

---

[11] Srđan Damjanović, "Nova rešenja akvizicionog sistema na stanicama za ispitivanje turbomlaznih motora sa smanjenom mernom nesigurnosti," *Doktorski rad*, Istočno Sarajevo, 2007, str. 79 do 83.

8.8.2 Jednačina modela

Jednačina modela mjerenja temperature *Tx* toplog spoja termopara koji se etalonira, kada je hladni spoj na 0 ºC, predstavljena je jednačinom

$$Tx=Ts + Cs \bullet \Delta Vs + Cs*\Delta Vr - \Delta Tos \bullet (Cs/Cso) + \Delta Th + \Delta Ta \qquad (8.28)$$

gdje su:

Ts (ºC) - Temperatura referentnog termopara u funkciji napona hladnog spoja, koji se nalazi na temperaturi 0 ºC.
ΔVs (μV) - Korekcija napona pri etaloniranju voltmetra.
ΔTos (ºC) - Korekcija mjerenja referentne temperature 0 ºC.
ΔTh (ºC) - Korekcija temperature zbog neravnomjerne raspodjele temperature u peći.
Cs (ºC/ μV) - Osjetljivost termopara S tip pri temperaturi 700 ºC.
Cso (ºC/ μV) - Osjetljivost termopara S tip pri temperaturi 0 ºC.
ΔTa (ºC) - Razlika u pokazivanju temperature referentnog termopara (ovo je komponenta mjerne nesigurnosti tip A).

Jednačina modela mjerenja napona *Vx* na krajevima termopara koji se etalonira, kada je hladni spoj na 0 ºC, predstavljena je jednačinom

$$Vx=Vs + \Delta Vr + \Delta T/Cx + \Delta Tox/Cxo + \Delta V_{LX} + \Delta Va \qquad (8.29)$$

gdje su:

Vs (μV) - Pokazivanje voltmetra.
ΔVr (μV) - Korekcija voltmetra zbog ograničene rezolucije.
ΔT (ºC) - Odstupanje temperature na kojoj se radi etaloniranje od temperature peći.
ΔTox - Korekcija mjerenja referentne temperature 0 ºC.
Cx (ºC/ μV) - Osjetljivost termopara K tip koji se etalonira pri temperaturi 700 ºC.
Cxo (ºC/ μV) - Osjetljivost termopara K tip koji se etalonira pri temperaturi 0 ºC.
ΔV_{LX} (μV) - Korekcija mjerenja napona zbog kompenzacionog kabla.
ΔVa (μV) - Razlika u pokazivanju pri mjerenju napona na nepoznatom termoparu (ovo je komponenta mjerne nesigurnosti tip A).

8.8.3 Kvantifikovanje komponenti mjerne nesigurnosti

Svaku komponentu mjerne nesigurnosti potrebno je detaljno opisati. Pri tome je potreno navesti iznose mjernih nesigurnosti svake komponente, koeficijent proširenja i procijenjenu vrstu raspodjele.

1) *Ts* (ºC) - Temperatura referentnog termopara u funkciji napona hladnog spoja na 0 ºC.

Termopar S tip koji se koristi kao etalon za mjerenje temperature u peći je etaloniran. Za mjerenje temperature od 700 ºC, sa etalonskim termoparom, u uverenju o etaloniranju data je proširena mjerna nesigurnost u iznosu od 0,75 ºC.

Ova proširena mjerna nesigurnost data je za normalnu raspodjelu rezultata mjerenja i koeficijent proširenja $k = 2$.

2) *Vs* (μV) - Pokazivanje digitalnog multimetra.

Proizvođač digitalnog multimetra HP 3458A je na mjernom opsegu 100 mV, koji se koristi za mjerenje napona na krajevima termopara, u svojoj dokumentaciji deklarisao grešku kao ±(0,3 μV od opsega + 0,0005% od očitanja). Tako da na temperaturi od 700 ºC prilikom mjerenja napona na krajevima termopara K tip, koji se etalonira, deklarisana greška mjerenja napona digitalnog multimetra iznosi ±0,45 μV. Pretpostavljeno je da rezultati mjerenja istosmjernog napona pomoću ovog multimetra imaju pravouganu raspodjelu.

3) *ΔVs* (μV) - Korekcija napona pri etaloniranju digitalnog multimetra.

Proizvođač digitalnog multimetra HP 3458A je na mjernom opsegu 100 mV, koji se koristi za mjerenje napona na krajevima termopara, u svojoj dokumentaciji deklarisao grešku kao ±(0,3 μV od opsega + 0,0005% od očitanja). Tako da na temperaturi od 700 ºC prilikom mjerenja napona na krajevima termopara S tip, koji se koristi kao etalonski termopar, deklarisana greška mjerenja napona digitalnog multimetra iznosi ±0,34 μV. Pretpostavljeno je da rezultati mjerenja istosmjernog napona pomoću ovog multimetra imaju pravouganu raspodjelu.

4) *ΔVr* (μV) - Korekcija digitalnog multimetra zbog ograničene rezolucije.

Proizvođač digitalnog multimetra je na opsegu 100 mV deklarisao najmanji značajan digit 0,01 μV. Iz ovoga slijedi da se svako pokazivanje digitalnog multimetra nalazi u granicama ± 0,005 μV. Pretpostavljeno je da uticaj zadnjeg digita na rezultat mjerenja pomoću ovog multimetra ima pravouganu raspodjelu.

5) *ΔTos* i *ΔTox* (ºC) - Korekcija mjerenja referentne temperature 0 ºC.

Proizvođač živinog termometra (koji se koristi za mjerenje referentne temperature 0 ºC) je deklarisao grešku kao ± 0,1 ºC. Pretpostavljeno je da rezultati mjerenja referentne temperature 0 ºC imaju pravouganu raspodjelu.

6) *ΔTh* (ºC) - Korekcija temperature zbog neravnomjerne raspodjele temperature u peći.

Prema dokumentaciji proizvođača homogenost temperature u vertikalnoj kalibracionoj peći je ±0,25 ºC za temperature do 800 ºC, a ± 0,5 ºC za temperature od 800 ºC do 1100 ºC. Pretpostavljeno je da homogenost temperature u peći ima pravouganu raspodjelu.

7) *Cs* (ºC/ μV) - Osjetljivost termopara S tip pri temperaturi 700 ºC.

Ova konstanta od 0,0952 ºC/ μV uzeta je iz tabele zavisnosti elektromotorne sile termopara S tip od temperature, pri temperaturi 700 ºC.

8) *Cso* (ºC/ μV) - Osjetljivost termopara S tip pri temperaturi 0 ºC.

Ova konstanta od 0,2 ºC/ μV uzeta je iz tabele zavisnosti elektromotorne sile termopara S tip od temperature, pri temperaturi 0 ºC.

9) *Cx* (ºC/ μV) - Osjetljivost termopara K tip pri temperaturi 700 ºC.

Ova konstanta od 0,0238 ºC/ μV uzeta je iz tabele zavisnosti elektromotorne sile termopara K tip od temperature, pri temperaturi 700 ºC.

10) *Cxo* (ºC/ μV) - Osjetljivost termopara K tip pri temperaturi 0 ºC.
Ova konstanta od 0,0256 ºC/ μV uzeta je iz tabele zavisnosti elektromotorne sile termopara K tip od temperature, pri temperaturi 0 ºC.

11) ΔTa (ºC) - Razlika u pokazivanju temperature referentnog termopara (komponenta mjerne nesigurnosti tip A).
Na svakoj mjernoj tački u kojoj se radi etaloniranja termopara K tip vrši se po deset mjerenja temperature na referentnom termoparu i termoparu koji se etalonira. Doprinos ove komponente mjerne nesigurnosti mjerenja temperature na referentnom termoparu računa se kao standardno odstupanje srednje vrijednosti. Za svako konkretno mjerenje dobija se druga komponenta mjerne nesigurnosti tip A. Ova mjerna nesigurnost se uzima sa normalnom raspodjelom.

12) ΔVa (μV) - Razlika u pokazivanju pri mjerenju napona na nepoznatom termoparu (komponenta mjerne nesigurnosti tip A).
Na svakoj mjernoj tački u kojoj se radi etaloniranja termopara K tip vrši se po deset mjerenja temperature na referentnom termoparu i termoparu koji se etalonira. Doprinos ove komponente mjerne nesigurnosti mjerenja napona na termoparu koji se etalonira računa se kao standardno odstupanje srednje vrijednosti. Za svako konkretno mjerenje dobija se druga komponenta mjerne nesigurnosti tip A. Ova mjerna nesigurnost se uzima sa normalnom raspodjelom.

13) ΔV_{LX} (μV) - Korekcija mjerenja napona zbog kompenzacionog kabla.
Termopar koji se etalonira kompenzacionim kablom se povezuje sa referentnom tačkom 0 ºC, odakle se bakarnim kablovima signal vodi do digitalnog multimetra. Kompenzacioni kabal K tip ispitan je u temperaturnom području od 0 ºC do 40 ºC. Kao rezultat ovog ispitivanja utvrđeno je da pad napona na kompenzacionom kablu između termopara i bakarnih kablova iznosi ±6 μV.

14) $\Delta T = T - Tx$ Odstupanje temperature na kojoj se radi etaloniranje od temperature peći.

Doprinos mjerne nesigurnosti ove komponente dobije se na osnovu procjene mjerne nesigurnosti temperature *Tx* toplog spoja termopara koji se etalonira, predstavljene u jednačini 8.28.

8.8.4 Mjerenje

U tabeli 8.6 prikazani su rezultati etaloniranja termopara K tip na 700 ºC u Metrološkoj laboratoriji ML-16 u VZ "ORAO". Etalonski termopar S tipa i termopar K tip koji se etalonira postavljeni su u kalibracionu peć jedan do drugaga na istu dubinu. Hladni krajevi termoparova spojeni su sa bakarnim kablovima u termos boci gdje se održava temperatura 0 ºC. Digitalnim multimetrom izvršeno je po deset mjerenja napona, koji se stvara na etalonskom termoparu S tip i na termoparu K tip koji se etalonira.

Tabela 8.6 Rezultati etaloniranja termopara K tip na 700 ºC

Termopar		S tip	K tip
Izmjerena vrijednost napona	1.	6311,94 µV	29266,16 µV
	2.	6313,74 µV	29279,24 µV
	3.	6315,38 µV	29271,68 µV
	4.	6314,24 µV	29268,31 µV
	5.	6313,59 µV	29264,54 µV
	6.	6312,68 µV	29259,12 µV
	7.	6312,17 µV	29257,53 µV
	8.	6311,73 µV	29251,64 µV
	9.	6311,24 µV	29248,82 µV
	10.	6310,67 µV	29242,79 µV
Srednja vrijednost napona		6312,738 µV	29260,983 µV
Srednja vrijednost temperature		703,55 ºC	703,14 ºC
Standardno odstupanje mjerenja napona za 10 mjerenja		1,47 µV	11,17 µV
Standardno odstupanje mjerenja temperature za 10 mjerenja		0,140 ºC	0,266 ºC
Standardno odstupanje srednje vrijednosti mjerenja napona		0,47 µV	3,53 µV
Standardno odstupanje srednje vrijednosti mjerenja temperature		0,044 ºC	0,084 ºC

8.8.5 Budžet nesigurnosti mjerenja temperature u peći *Tx*

Na osnovu jednačine 8.28 modela mjerenja temperature *Tx* toplog spoja termopara koji se etalonira, napravljen je budžet doprinosa pojedinih komponenti mjernih nesigurnosti prilikom mjerenja temperature u peći sa etalonskim termoparom S tip. Ovaj budžet mjernih nesigurnosti prikazan je u tabeli 8.8.

8.8.6 Budžet nesigurnosti mjerenja napona na krajevima termopara

Na osnovu jednačine 8.29 modela mjerenja napona *Vx* na krajevima termopara, koji se etalonira, kada je hladni spoj na 0 ºC, napravljen je budžet doprinosa pojedinih komponenti mjernih nesigurnosti prilikom mjerenja napona na krajevima termopara sa digitalnim multimetrom HP3458A. Ovaj budžet mjernih nesigurnosti prikazan je u tabeli 8.8.

Tabela 8.7 Budžet mjernih nesigurnost mjerenja temperature u peći

Veličina	Procjena	Standardna nesigurnost	Vjerovatnoća raspodjele	Koeficijent osjetljivosti	Doprinos nesigurnosti
Ts	703,55 ºC	0,325 ºC	normalna	1	0,325 ºC
ΔVs	0 μV	$(0{,}34/\sqrt{3})$ μV	pravougaona	0,0952 ºC/μV	0,019 ºC
ΔVr	0 μV	$(0{,}005/\sqrt{3})$ μV	pravougaona	0,0952 ºC/μV	0,00027 ºC
ΔTos	0 ºC	$(0{,}1/\sqrt{3})$ ºC	pravougaona	0,476	0,027 ºC
ΔTh	0 ºC	$(0{,}25/\sqrt{3})$ ºC	pravougaona	1	0,144 ºC
ΔTa	0 ºC	0,044 ºC	normalna	1	0,044 ºC
Cs	0,0952 ºC/ μV		konstanta		
Cso	0,2 ºC/μV		konstanta		
Cs/Cso	0,476		konstanta		
Tx	703,55 ºC		normalna		0,36 ºC

Tabela 8.8 Budžet mjernih nesigurnosti mjerenja napona Vx na krajevima termopara

Veličina	Procjena	Standardna nesigurnost	Vjerovatnoća raspodjele	Koeficijent osjetljivosti	Doprinos nesigurnosti
Vs	29260,983 μV	$0{,}45/\sqrt{3}$ μV	pravougaona	1	0,26 μV
ΔVr	0 μV	$(0{,}005/\sqrt{3})$ μV	pravougaona	0,0952 ºC/μV	0,0029 μV
ΔT	0,4 ºC	(0,36) ºC	normalna	42 μV/ºC	15,12 μV
ΔTox	0 ºC	$(0{,}1/\sqrt{3})$ ºC	pravougaona	39 μV/ºC	2,25 μV
ΔV_{LX}	0 μV	$(6/\sqrt{3})$ μV	pravougaona	1	3,47 μV
ΔVa	0 ºC	3,53 μV	normalna	1	3,53 μV
Cx	0,0238 ºC/ μV		konstanta		
1/ Cx	42 μV/ºC		konstanta		
Cxo	0,0256 ºC/ μV		konstanta		
1/ Cxo	39 μV/ºC		konstanta		
Vx	29277,95 μV		normalna		16,07 μV

8.8.7 Rezultat etaloniranja

Proširena mjerna nesigurnost pridružena mjerenju temperature peći je:

$U = k \bullet u(Tx) = 2 \bullet 0{,}36$ ºC = 0,72 ºC .

Proširena mjerna nesigurnost pridružena elektromotornoj sili termopara koji se etalonira je:

$U = k \bullet u(Vx) = 2 \bullet 16{,}07$ μV = 32,14 μV.

Rezultat etaloniranja termopara K tip na temperaturi 700 ºC, pri čemu je njegov referentni spoj bio na 0 ºC, je elektromotorna sila koja se javila na njegovim krajevima od:

29277,95 μV ± 32,14 μV.

Navedena proširena mjerna nesigurnost mjerenja napona na krajevima termopara koji se etalonira, kada je hladni spoj na 0 ºC utvrđena je kao standardna mjerna nesigurnost pomnožena sa koeficijentom proširenja k = 2, koji za normalnu raspodjelu odgovara nivou povjerenja od približno 95%.

8.9 ZNAČAJ I DOPRINOS MJERNE NESIGURNOSTI

Mjerna nesigurnost je kvantitativni pokazatelj rezultata mjerenja. Ona daje odgovor na pitanje koliko dobro rezultat predstavlja vrijednost mjerene veličine i omogućava korisniku rezultata da ocijeni njegovu pouzdanost. Poznavanje mjerne nesigurnosti omogućuje poređenje rezultata mjerenja dobijenih iz različitih izvora, različitim metodama, u različitim laboratorijama i tako pomaže smanjenju trgovačkih barijera. Poznavanje mjerne nesigurnosti je značajno i presudno za ocijenu usaglašenosti, kada se rezultat poredi sa graničnim vrijednostima definisanim u specifikacijama ili propisima.

Mjerna nesigurnost je značajna za:

- kontrolu kvaliteta i obezbjeđenje kvaliteta u proizvodnji,
- usklađivanje sa zakonom i zakonskim regulativama,
- sprovođenje osnovnih istraživanja i primjenu istraživanja i razvoja u nauci i praksi,
- etaloniranje etalona i mjerila i sprovođenje ispitivanja u okviru nacionalnog metrološkog sistema, sa ciljem da se postigne sljedljivost do nacionalnih, odnosno međunarodnih etalona,
- razvoj, održavanje i poređenje međunarodnih i nacionalnih referentnih etalona uključujući i referentne materijale,
- akreditaciju laboratorija za etaloniranje i ispitivanje.

L I T E R A T U R A

[1] VEE PRO 8.5. Help, Agilent Technologies, 2007.

[2] VEE PRO 6.0. Help, Agilent Technologies, 2000.

[3] Agilent Technologies Libraries for Instrument Control Revision: L.02.00, Agilent Technologies CD, 2002.

[4] Microsoft Visual Basic 6.0 Help, Microsoft, 1999.

[5] Dr Jozo J. Dujmović, *Programski jezici i metode programiranja*, Akademska misao, Beograd, 2003.

[6] Dr Tihomir Latinović, *Osnove programiranja (Visual Basic)*, Biblioteka Informacione tehnologije, Banja Luka, 2007.

[7] Dr Lazar Miličević, Mr Lazar Radovanović, *Programiranje (Visual Basic)*, Ekonomski fakultet, Brčko, 2005.

[8] Srđan Damjanović, Predrag Katanić, Borislav Drakul, *Zbirka zadataka iz poslovne informatike*, Fakultet spoljne trgovine, Bijeljina, 2008.

[9] Command Reference Manual HP3852A Data Acquisition/Control Unit, Hewlettt Packard Company, 1987 .

[10] User's Guide Agilent 3497A Data Acquisition/Switch Unit, 1999.

[11] User's Guide Agilent 82357A USB/GPIB Interface for Windows, Agilent Technologies, 2000.

[12] Srđan Damjanović, "Razvoj i etaloniranje akvizicionog sistema na stanici za ispitivanje turbo-mlaznih motora," *Magistarski rad*, Beograd, 2004.

[13] Srđan Damjanović, "Nova rešenja akvizicionog sistema na stanicama za ispitivanje turbomlaznih motora sa smanjenom mernom nesigurnosti," *Doktorski rad*, Istočno Sarajevo, 2007.

[14] Dragan Stanković, *Fizičko tehnička merenja - senzori*, Naučna knjiga, Beograd, 1997.

[15] Dragan Stanković, Predrag Osmokraković, *Praktikum laboratorijskih vežbi iz fizike*, Zavod za fiziku tehničkih fakulteta u Beogradu, 2005.

[16] Petar Pravica, Ivan Bagarić, *Metrologija električnih veličina opšti deo*, Nauka, Beograd, 1993.

[17] Guide to the Expression of the Uncertainty in Measurement, ISO, 1993.

[18] EA-4/02 Expresion of the Uncertainty of Measurement in Calibration, *European co-operation for Accreditation*, 1999.

[19] EA-10/08 Calibration of Thermocouples, *European co-operation for Accreditation*, 1997.

[20] Ascii kod, http://frank.harvard.edu/aoe/images/t10r3.pdf, 28.01.2011.

“ORAO” A.D. Bijeljina

OSNOVNE DJELATNOSTI

- Pregledi, održavanje prema tehničkom stanju, generalni remont i ispitivanje turbomlaznih motora, njihovih podsklopova i agregata;
- Proizvodnja, montaža i testiranje turbomlaznih motora i drugih sklopova i dijelova u oblasti vazduhoplovstva i u oblasti mašinske industrije;
- Laboratorijska kalibracija u području električnih, dimenzionalnih i mehaničkih veličina kao i laboratorijsko ispitivanje sastava i fizičko-hemijskih karakteristika materijala.

Laboratorije predstavljaju značajan potencijal ORLA.Tri su osnovne djelatnosti kojima se laboratorije bave:

- metrologija električnih i neelektričnih veličina,
- ispitivanja materijala sa razaranjem i bez razaranja,
- razvojna ispitivanja i projektovanja laboratorijskih kapaciteta.

Metrološka laboratorija posjeduje:

- Sertifikat *BAS EN ISO/IEC 17025:2006, LK-18-01*, izdat od Instituta BiH za Akreditiranje kojim Metrološka laboratorija ispunjava zahtjeve u pogledu osposobljenosti za obavljanje kalibracija u području električnih, dimenzionalnih i mehaničkih veličina;
- Sertifikat za etaloniranje Metrološke Laboratorije ML16 izdat od Ministarstva odbrane Republike Srbije.

“ORAO” A.D. za proizvodnju i remont, Bijeljina,
Šabačkih đaka bb, 76300 Bijeljina, Republika Srpska/ BiH
Tel. Direktor: +387 (0)55 20 20 04,
Tel./Fax. +387 (0)55 20 20 07,
Tel. Laboratorija +387 (0)55 20 20 06,
Tel. marketing +387 (0)55 20 20 03,
e-mail: marketing@orao.aero

Predstavništvo **‘’ORAO’’ d.o.o. Beograd**,
Bulevar Dr Zorana Đinđića br. 98A,
11000 Beograd, Srbija
Tel. +381 (0)11 311 05 87,
Fax. +381 (0)11 311 05 88,
e-mail: orao@beocity.net,
orao@datanet.rs

IBIS INSTRUMENTS je osnovan 1996. godine sa ciljem da ispuni rastuće zahteve za konkurentnim proizvodima i sistemima namenjenim efikasnom održavanju, otklanjanju problema, nadzoru i upravljanju telekomunikacionim mrežama. Danas, Ibis Instruments predstavlja organizovanu i specijalizovanu firmu koja može da ponudi kompletan portfolio test i merne opreme i OSS rešenja u saradnji sa svojim partnerima, vodećim svetskim proizvođačima. Pored distribucije i implementacije vrhunskih tehnoloških rešenja Ibis Instruments pruža i sve vrste:

Pre-sale usluga - konsalting, prezentacije i demonstracije opreme i sistema, seminari i predavanja

After-sale usluga - tehnička podrška, edukacija korisnika, inženjering, projektovanje, servisiranje, redovno održavanje opreme i kalibracija.

T&M REŠENJA i PROIZVODI iz programa Ibis Instruments ispunjavaju zahteve korisnika za merenjem, inspekcijom, monitoringom, planiranjem, optimizacijom, kontrolom kvaliteta servisa i upravljanjem.

Adresa: Goce Delčeva 46, 11070 Novi Beograd
Telefoni: + 381 11 319 3157 i + 381 11 260 7502
Telefaks: + 381 11 319 3406
E-mail: info@ibis-instruments.com

CIP – Каталогизација у публикацији
Народна и универзитетска библиотека
Републике Српске, Бања Лука

004.432.2VEE Pro

ДАМЈАНОВИЋ, Срђан
Programski jezik VEE Pro / Srđan Damjanović, Predrag Katanić. - Istočno Sarajevo : Elektrotehnički fakultet, 2011 (Bijeljina : Grafika Gole). - 197 str. : ilustr. ; 25 cm

Tiraž 200. - Napomene i bibliografske reference uz tekst. - Bibliografija: str. 197.

ISBN 978-99938-624-7-5
1. Катанић, Предраг [аутор]

COBISS.BH-ID 2152728