Innovative Teaching Approaches in development of Software Designed Instrumentation and its application in real-time systems

Practicum of measurement and data acquisition systems

Integral presentation for LabVIEW environment

Co-funded by the Erasmus+ Programme of the European Union

Innovative Teaching Approaches in development of Software Designed Instrumentation and its application in real-time systems

Faculty of Technical Sciences

Ss. Cyril and Methodius University
Faculty of Electrical Engineering and Information Technologies

Zagreb University of Applied Sciences

School of Electrical Engineering
University of Belgrade

Faculty of Physics
Warsaw University of Technology

Co-funded by the Erasmus+ Programme of the European Union

Itasdi

The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Innovative Teaching Approaches in development of Software Designed Instrumentation and its application in real-time systems

Praktikum iz merno-akvizicionih sistema

Integralna prezentacija za LabVIEW okruženje

Co-funded by the Erasmus+ Programme of the European Union

PROGRAMSKI PAKET LabVIEW

PREGLED KURSA

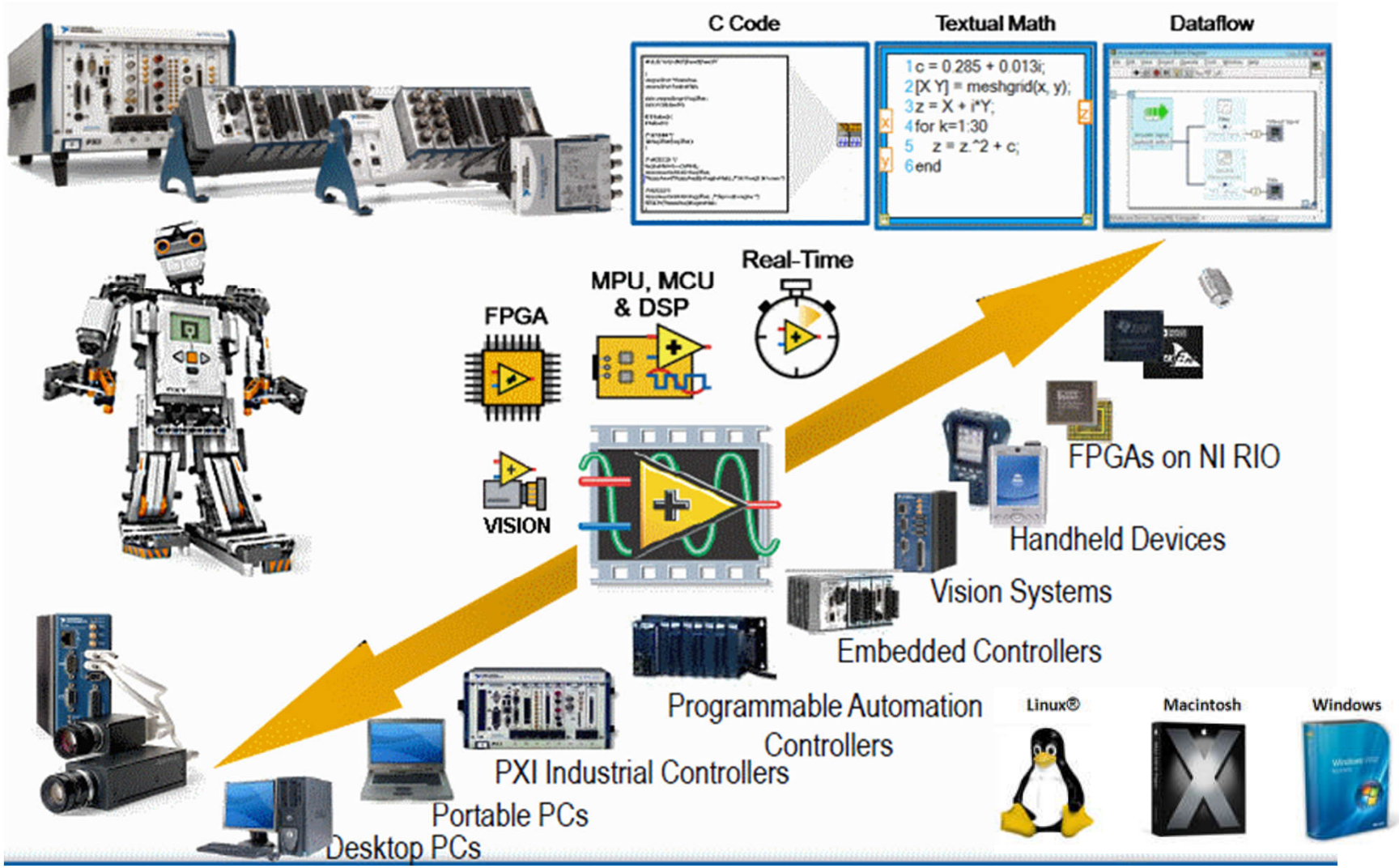
- Upoznavanje sa LabVIEW-om
  - Karakteristike, Block Diagram, Front Panel,
- Dataflow način programiranja
  - Realizacija jednostavnog programa
- Debagiranje
- Modularnost
- Petlje
- Nizovi, klasteri, grafičko prikazivanje podataka
- Strukture grananja
  - if…then…else, case…
  - Event Driven programiranje
- Stringovi
- Rad sa fajlovima
- Programska kontrola korisničkog interfejsa
- Sekvencijalno programiranje i mašina stanja
- Lokalne promenljive
- Komunikacija između petlji (race condition), redovi (queue)
- Šabloni programiranja
- Akvizicija i generisanje signala
- Priprema projekta za distribuciju
- Distribucija aplikacije

LabVIEW - Laboratory Virtual Instrumentation Engineering Workbench

- **LabVIEW omogućava:**
  - **razvoj uređaja i sistema i njihovo testiranje,**
  - **realizaciju akvizicije i obrade mernih rezultata,**
  - **izradu celokupnih kontrolnih sistema (SCADA).**
- **Jednostavna i efikasna izrade cele aplikacije.**
- **Sveprisutan.**

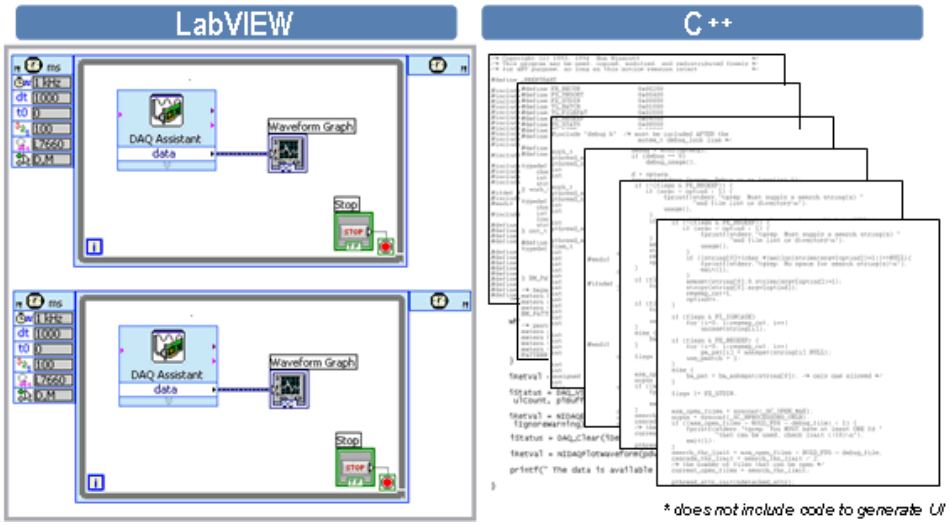
LABVIEW

- **Podržan na velikom broju platformi.**
- **Lako se povezuje sa brojnim programskim jezicima.**

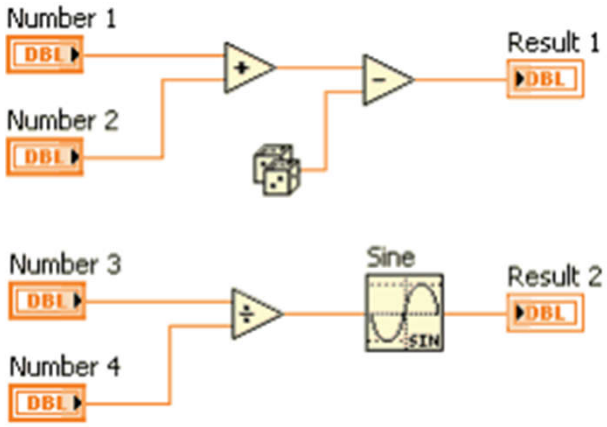


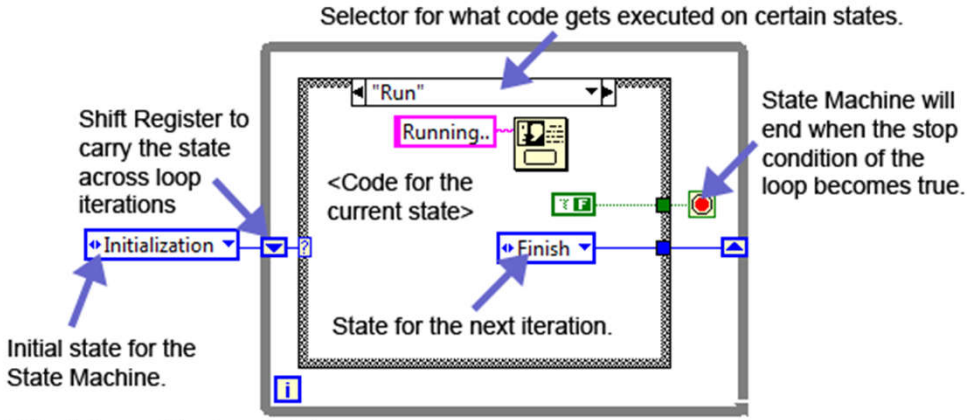
LABVIEW - Uvod

Paralelno izvršavanje.



Dataflow programming



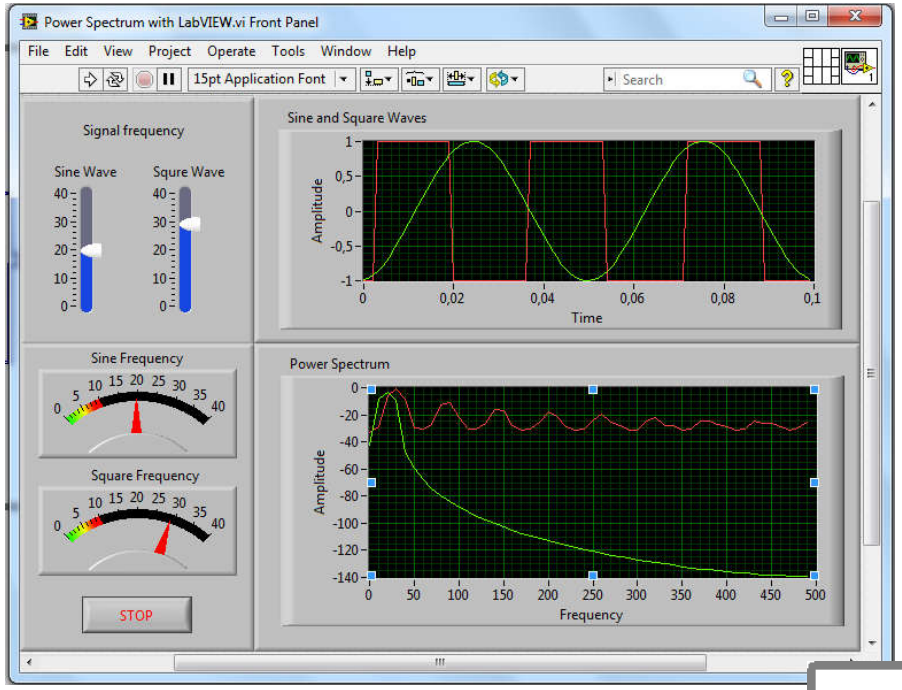


This state machine has 3 states.
1. Initialization
2. Run
3. Finished

State machine.

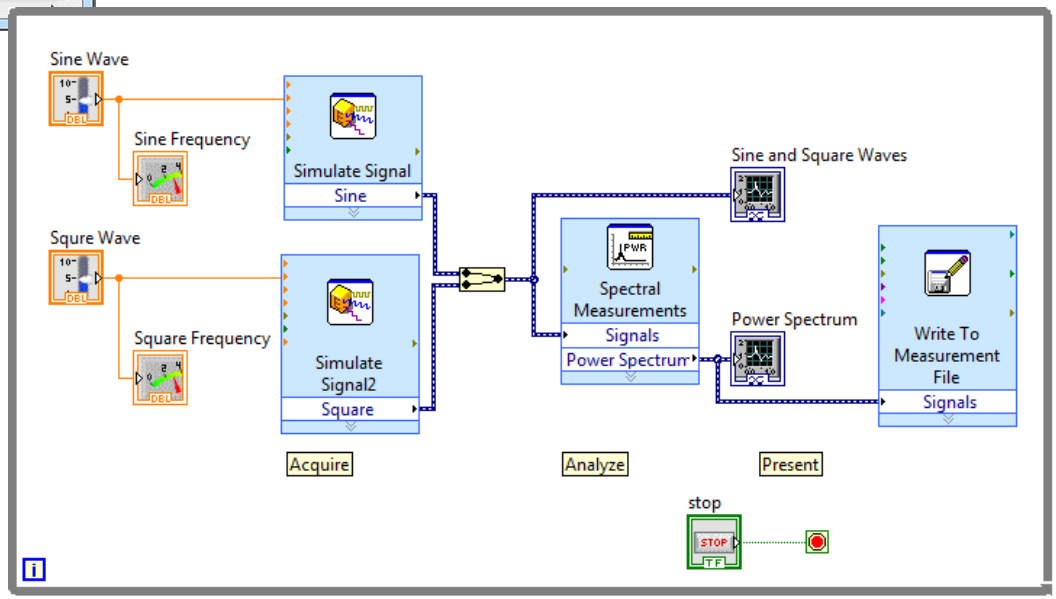
The typical variable used for a state machine is the “Enum”. This allows you to a set have predefined values, and so it more accurate than using text/strings.

LABVIEW - Uvod

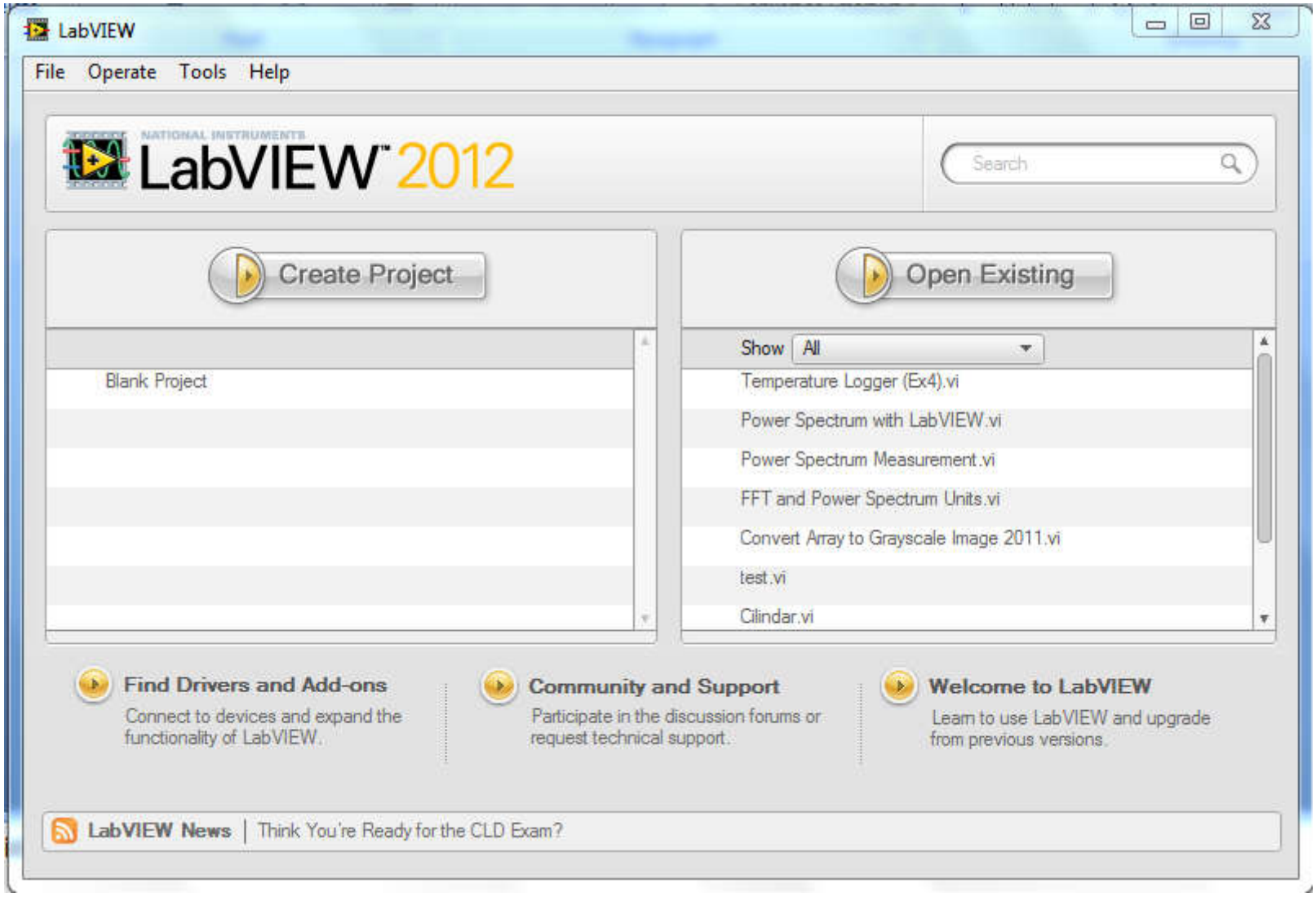


- *Front panel* – interakcija korisnika sa aplikacijom:
  - Indikatori – prikaz rezultata obrade,
  - Kontrole – zadavanje parametara.

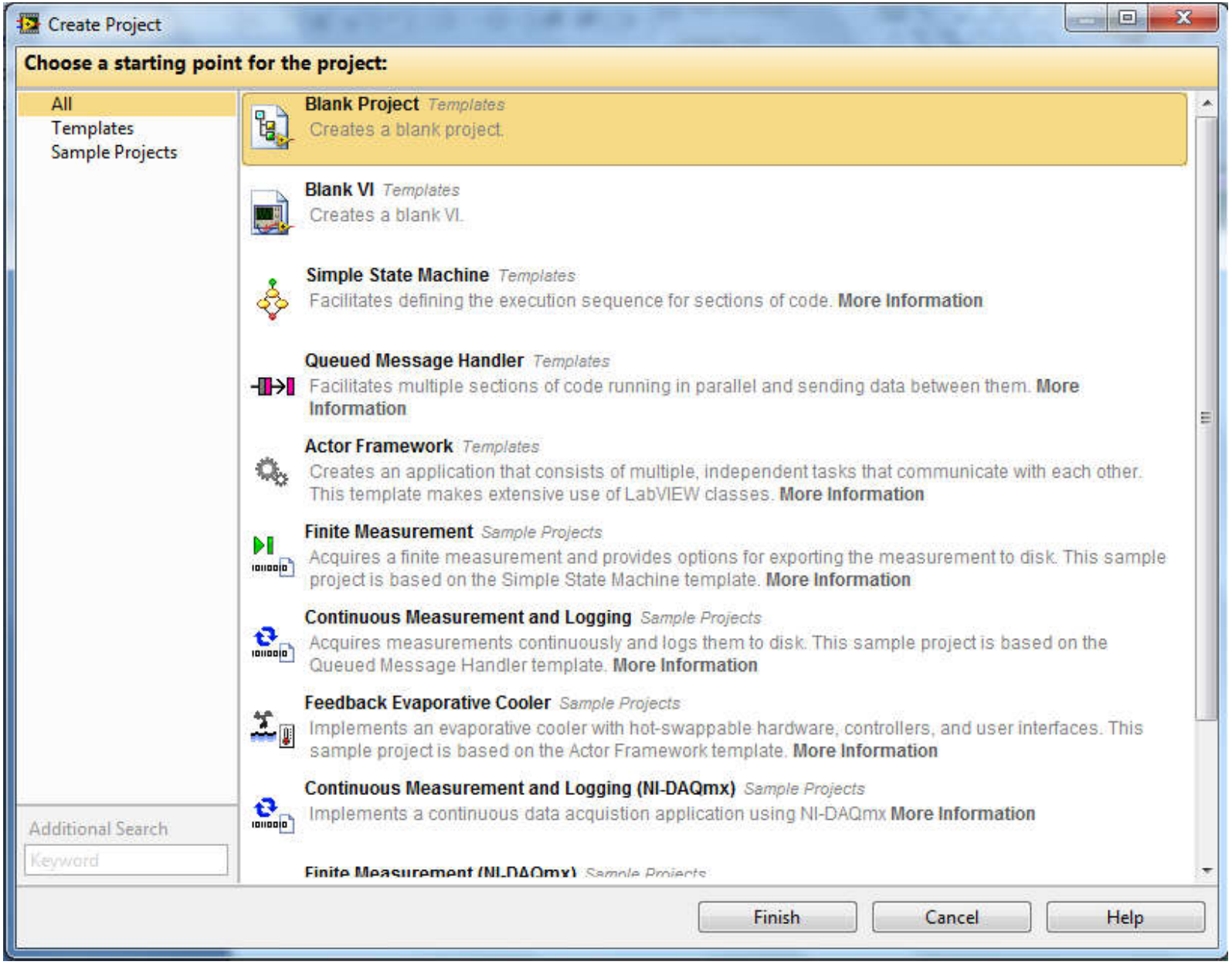
- *Blok diagram* – programski kod aplikacije.



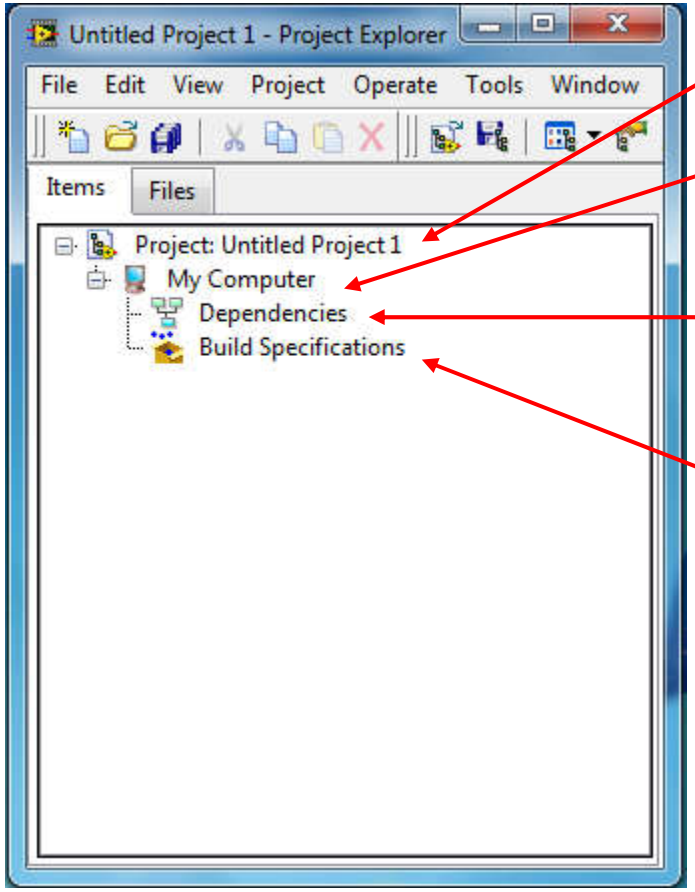
LabVIEW - pokretanje



LabVIEW - pokretanje



LabVIEW - pokretanje



Project Explorer

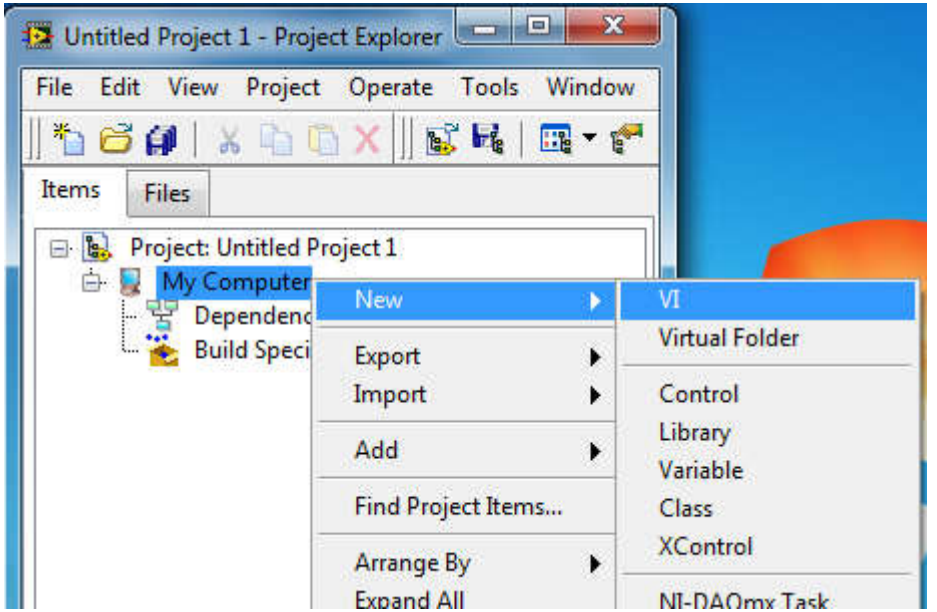
Project root – sadrži sve komponente projekta, ime odgovara imenu projekta.

My computer – platforma za izvršavanje projekta.

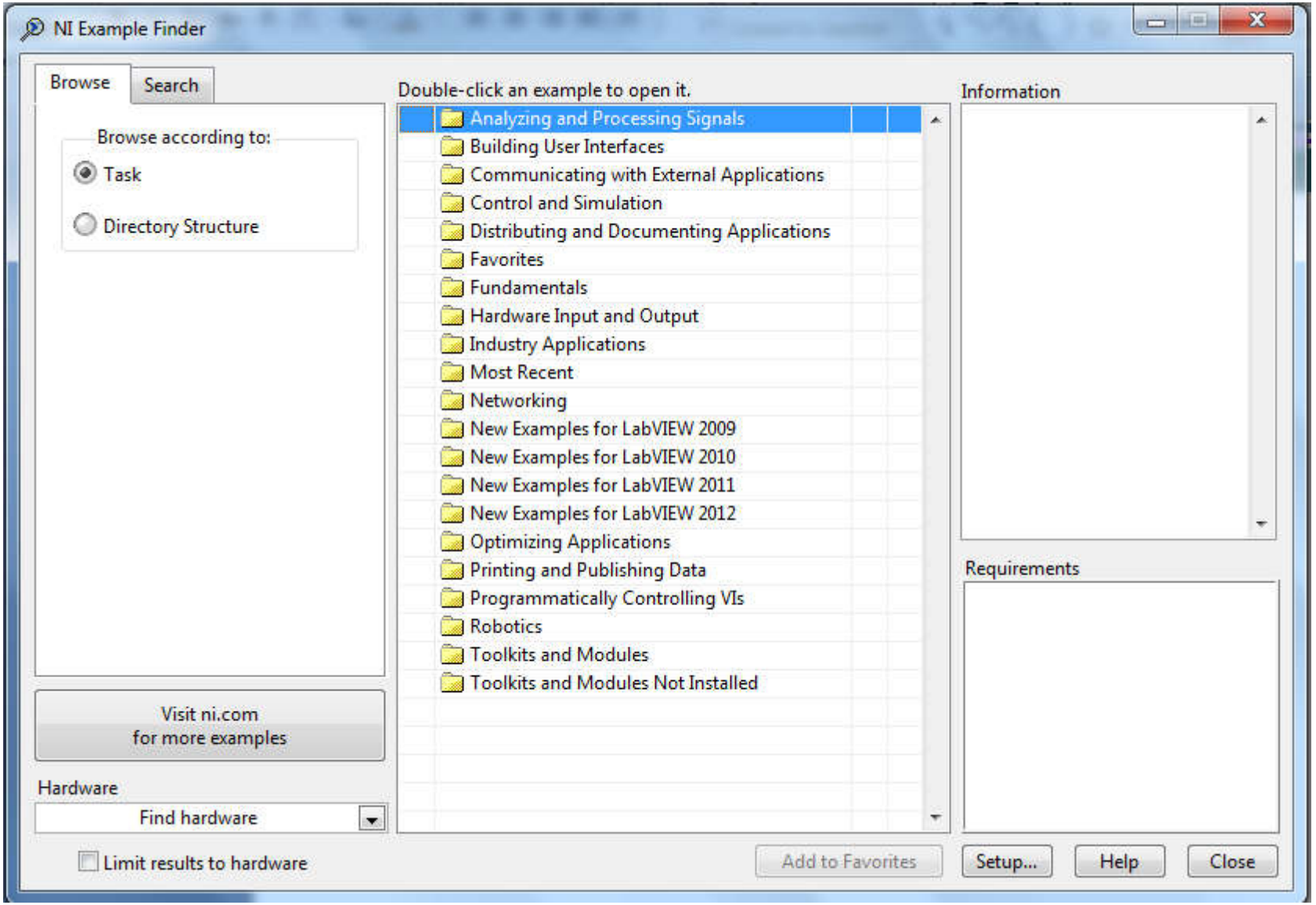
Dependencies – Sadrži VI-eve i druge fajlove koji su potrebni za izvršavanje projekta na zadatoj platformi.

Build Specification – Konfiguracija za kreiranje exe verzije, *stand alone* aplikacije (*installer*-a), i drugih distribucija projekta.

Add New *VI – Virtual Instrument*

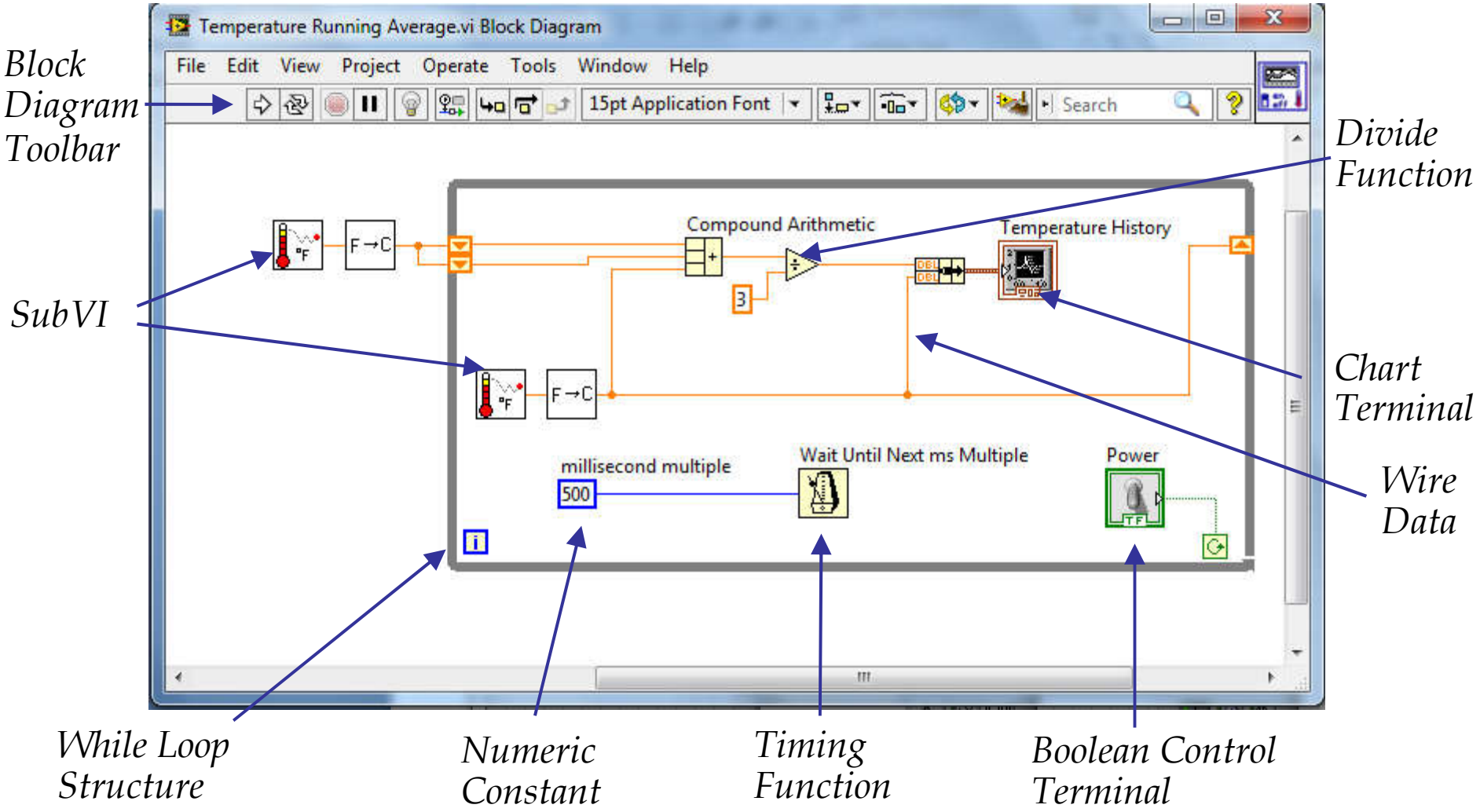


LabVIEW – novi *project*



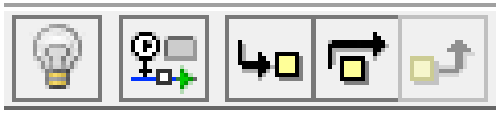
Korišćenje postojećih primera – *Help\Find Examples*

Block Diagram (BD) Window



Front Panel and Block Diagram Toolbars

Run button
Continuous Run button
Abort button
Pause/Continue button



Additional Buttons on the Block Diagram Toolbar

- Execution Highlighting button
- Retain Wire Values
- Step Into button
- Step Over button
- Step Out button

Font ring
Alignment ring
Distribution ring
Resize ring
Reorder ring
Clean Up Diagram
Context Help Button

 Warning indicator

 Enter button

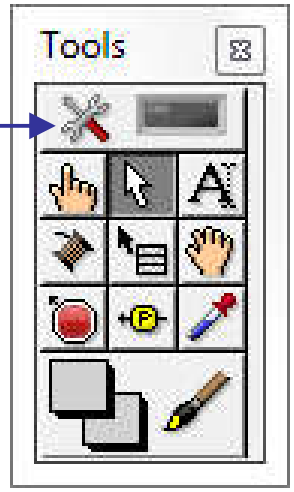
 Broken Run button

Tools Palette – Paleta alata

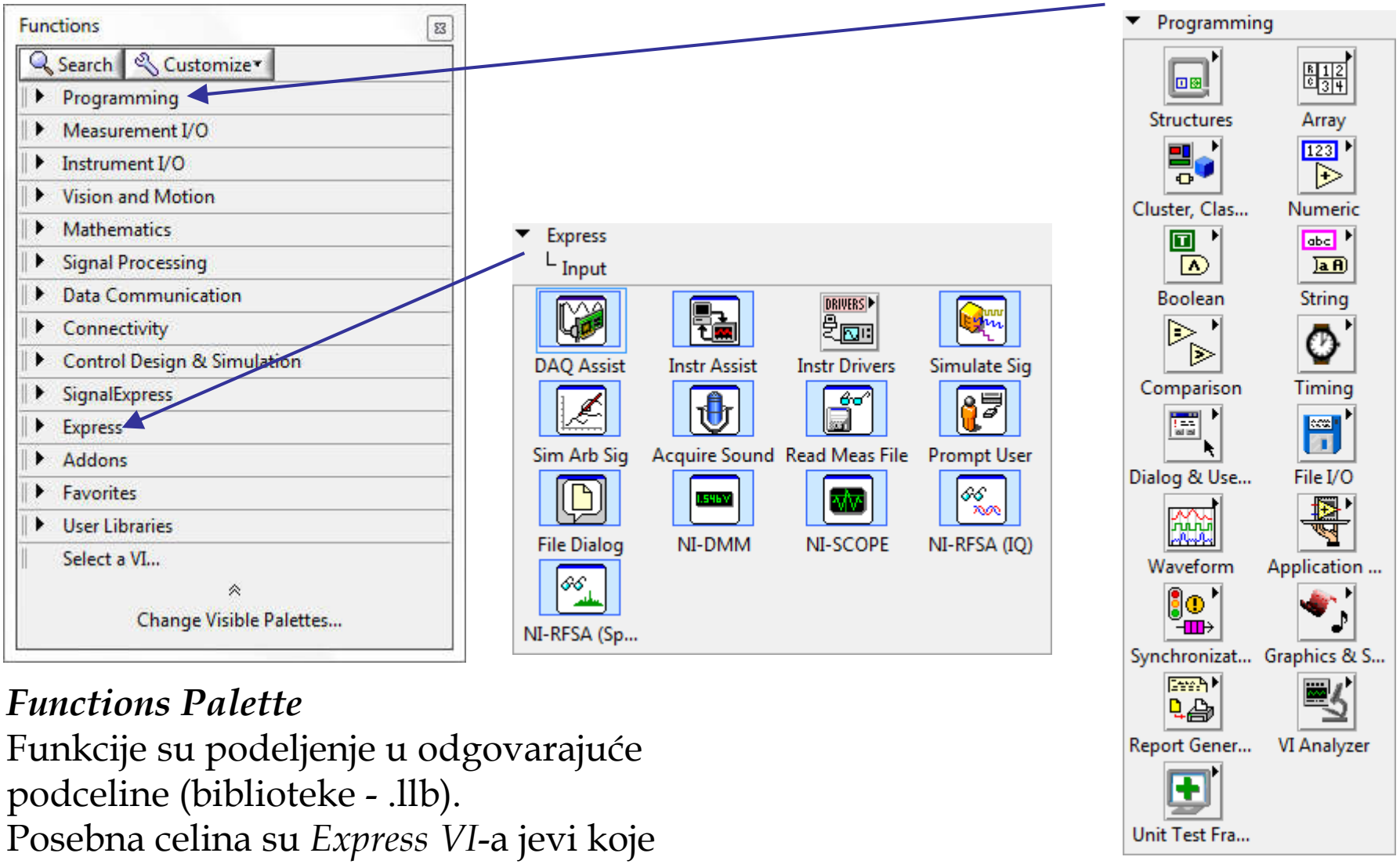
- Alat predstavalja specijalni mod funkcionisanja "miša".
- LabVIEW može automatski da izabere alat koji mu je potreban.
- Dostupan je i na *Front Panel*-u i na *Block Diagram*-u, i služi za rad i modifikovanje objekata na njima.
- Da bi se prikaza *Tool Palette* potrebno je ***View » Toops Palete*** ili ***Shift + Right mouse click*** za trenutni prikaz.

Automatska selekcija alata uključena

Automatska selekcija alata isključena



Block Diagram – Functions Palette



Functions Palette

Funkcije su podeljenje u odgovarajuće podceline (biblioteke - .llb).

Posebna celina su *Express VI*-a jevi koje omogućavaju jednostano podešavanje pametara funkcije uz pomoć posebnog *UI (User Interface)*.

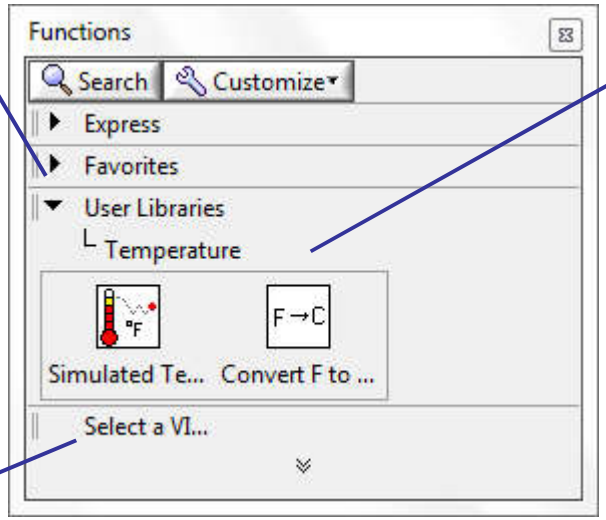
Block Diagram – Functions Palette

User Libraries

Temperature – korisnička bibloteka – potreno je iskopirati .llb *file* (*Library File in LabVIEW*) u folder C:\..\National Instruments\LabVIEW x\user.lib (gde x predstavlja verzija LabVIEW)

Favorites

Moguće je dodati bilo koju funkciju ili biblioteku.



Kreiranje .lbb

- *Tools » LLB Manager*

Promena imena na *Functions Pallete*

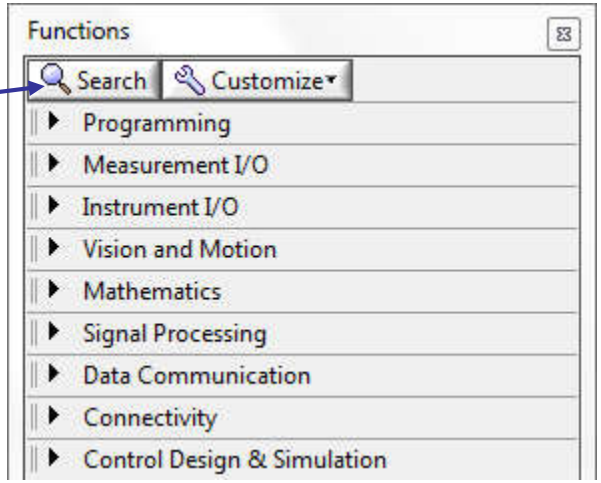
- *Tools » Advanced » Edit Palette Set...*

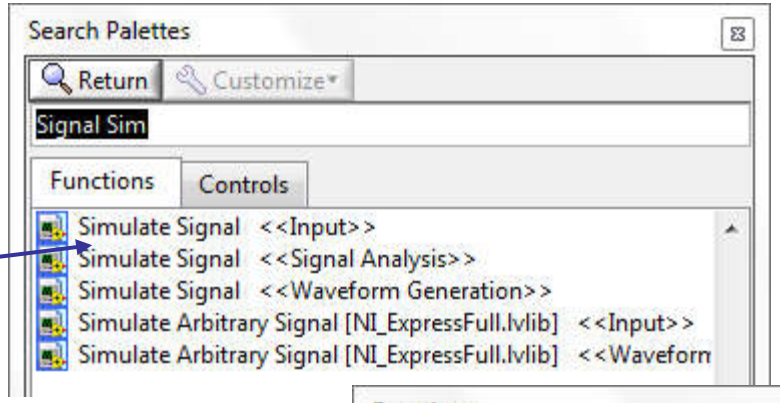
Select a VI...

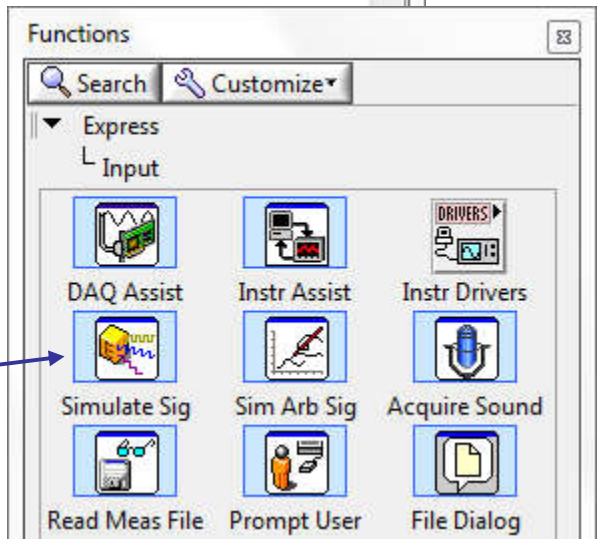
Korišćenje postojećeg VI-a kao subVI-a.

Block Diagram – Functions Palette

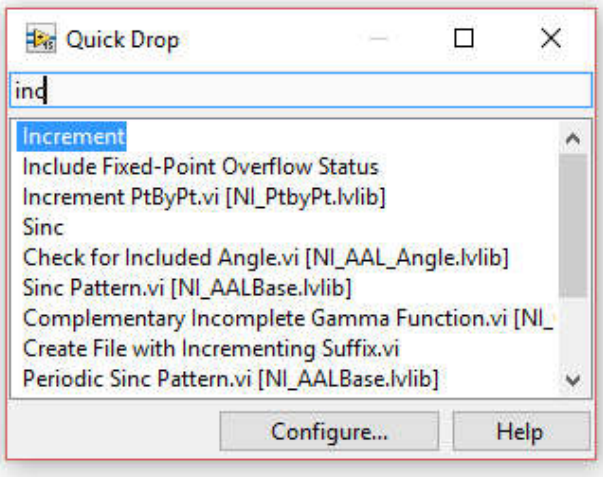
Pronalaženje funkcije
npr. *Simulate Signal*







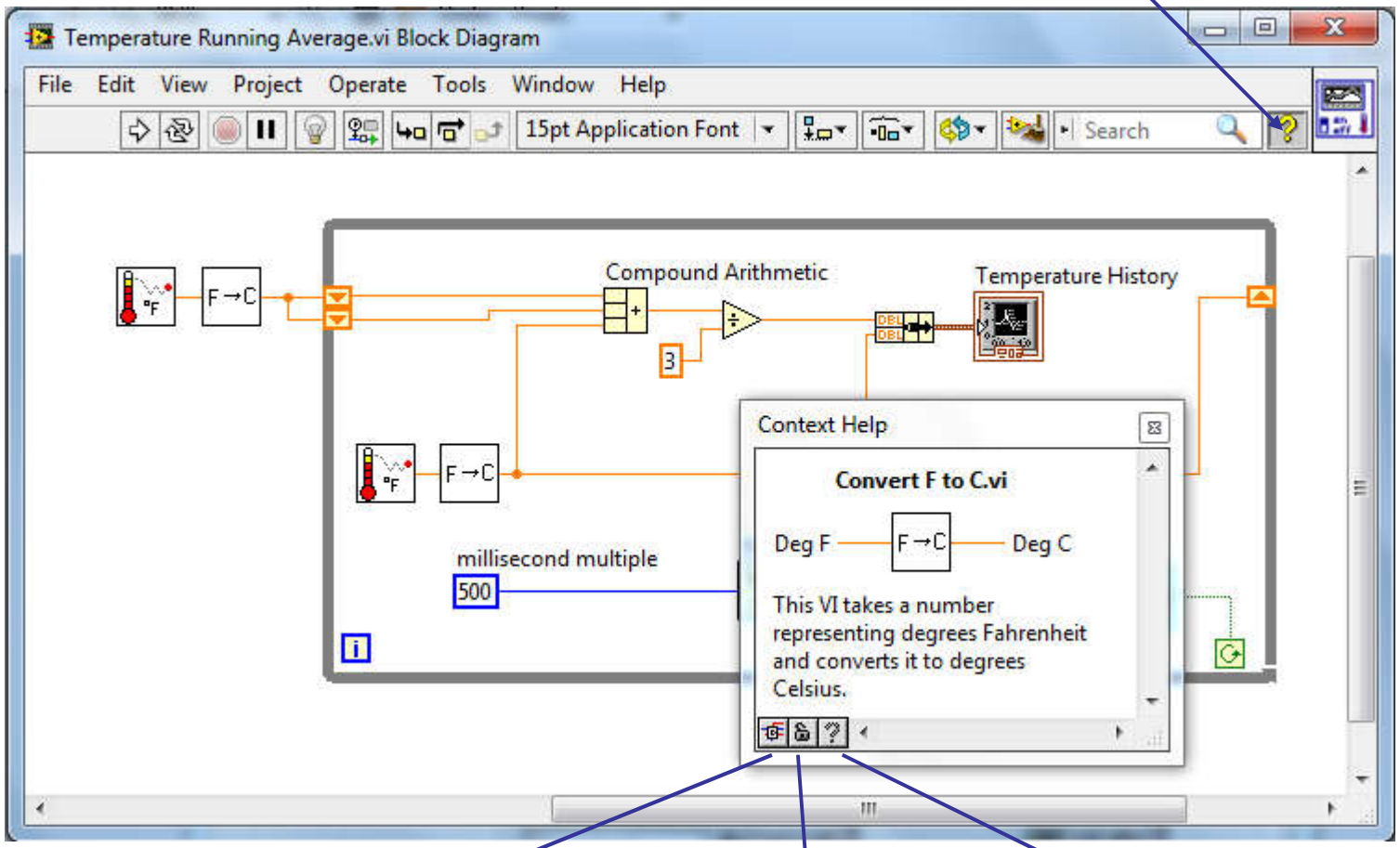
Quick Drop – brzo dodavanje funkcija na *BD* (Ctrl + Space)



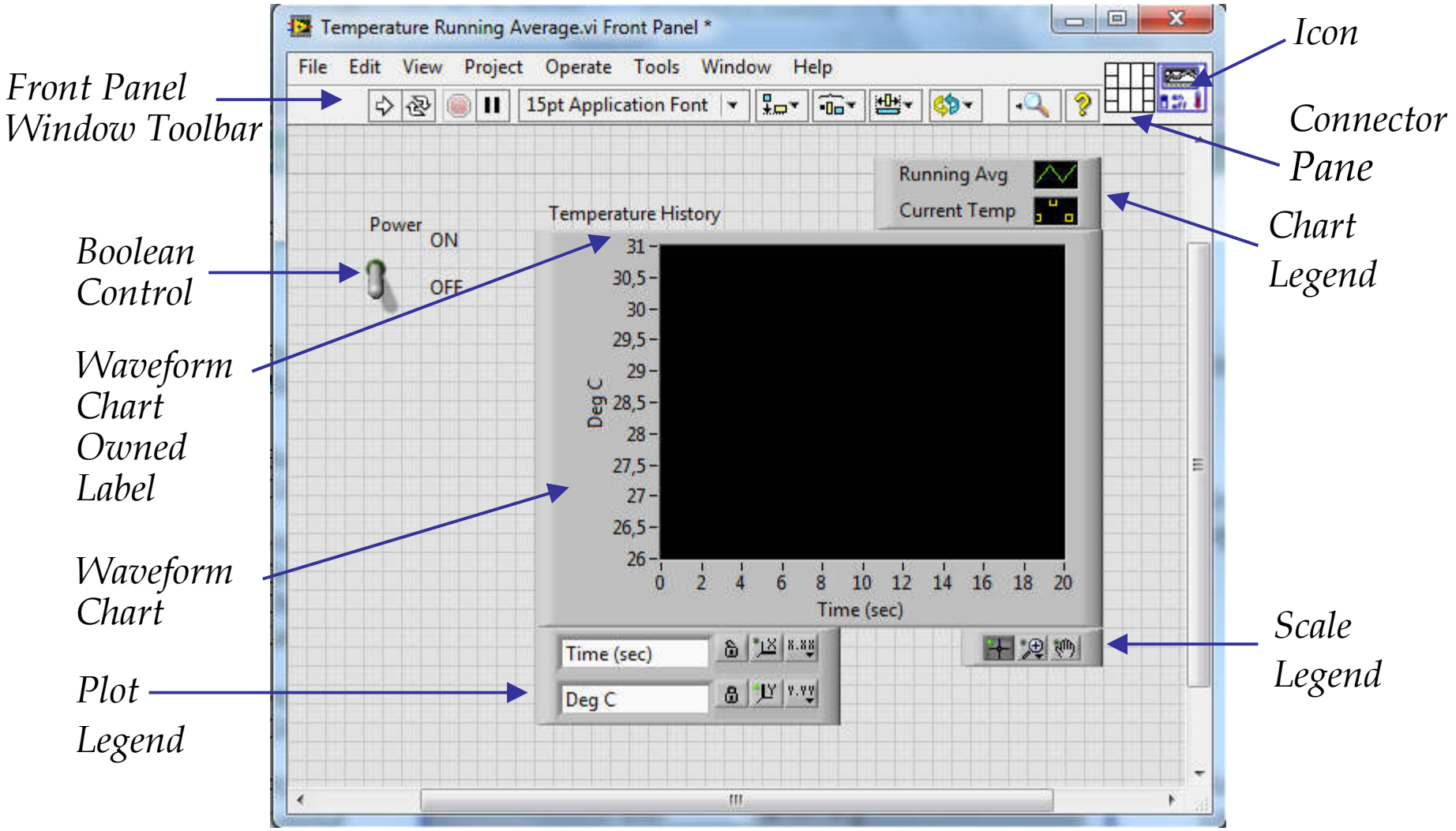
Block Diagram – Context Help

Context Help

- *Description* za *subVI* ili function node iznad koga se trenutno nalazi *Mouse Pointer*
- Aktiviranje: 1. *Ctrl + H*, 2. *Help » Show Context Help*, 3. 



Front Panel (FP) Window

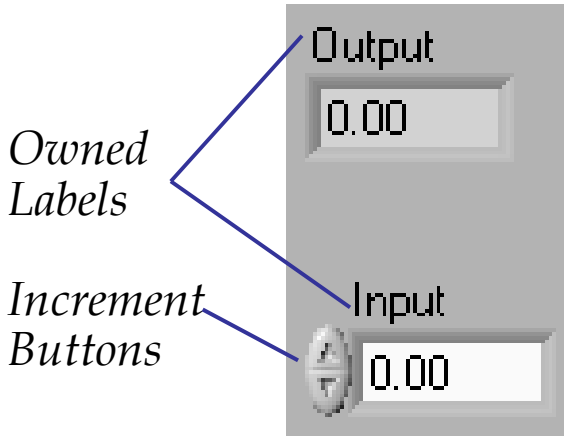


Kreiranje *Front Panel*-a

- *Front panel* se sastoji od kontrola (ulaza) i indikatora (izlaza).
- Izbora kontola/indikatora sa *Controls Palette*.

Boolean Control

Boolean Indicator

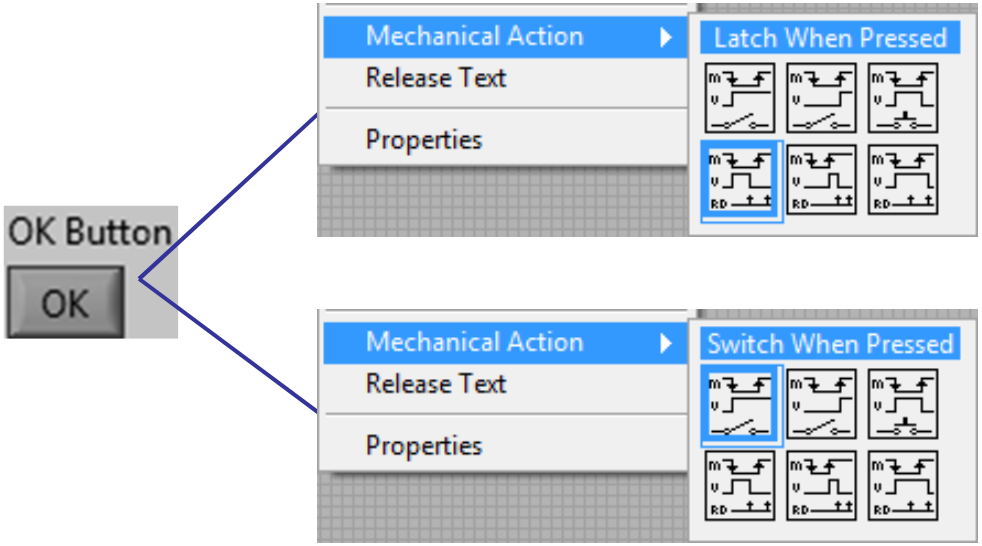


Owned Labels

Increment Buttons

Numeric Indicator

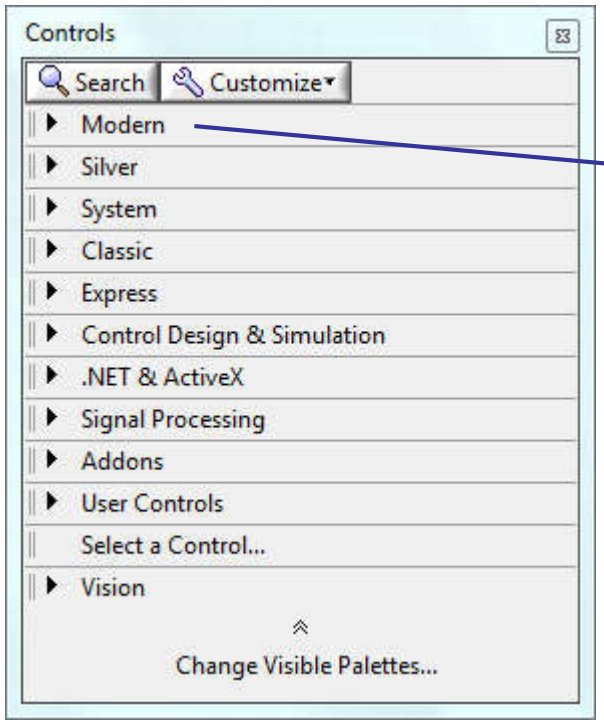
Numeric Control

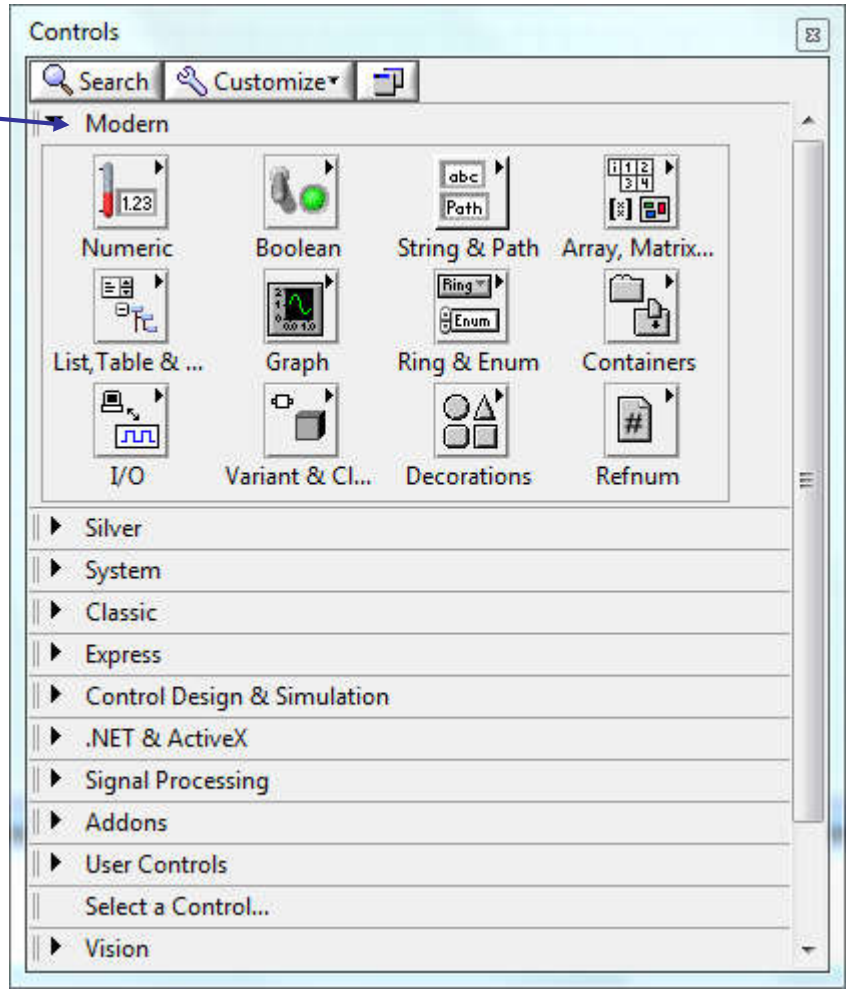


Latch – ostaje u stanju 1 sve dok LabVIEW ne pročita stanje kontrole, nakon čega se vraća u stanje 0.

Switch – ostaje u stanju 1 sve korisnik ne promeni stanje i vrati kontrolu u stanje 0.

Front Panel – Controls Palette



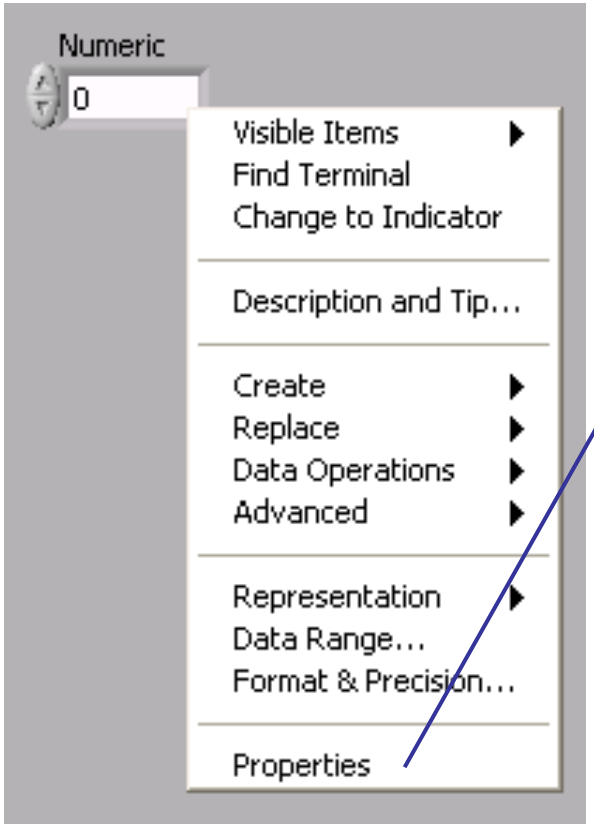


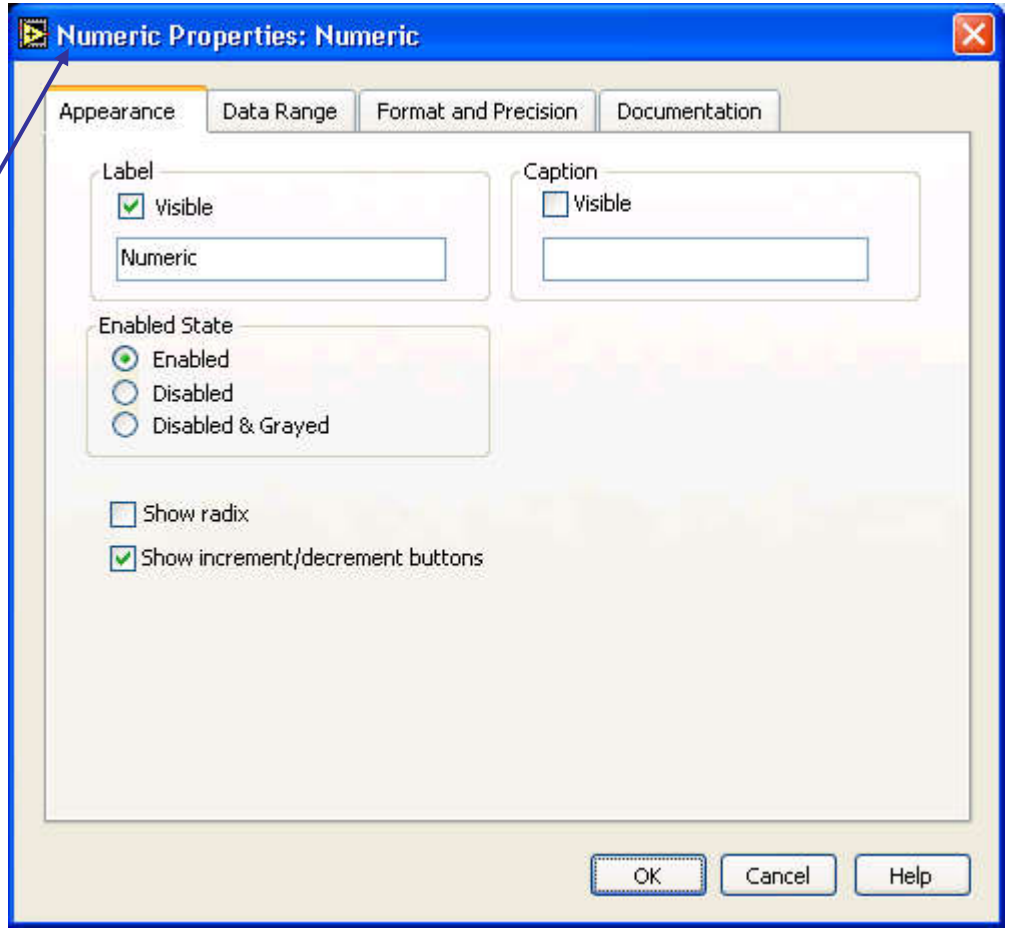
Controls Palette

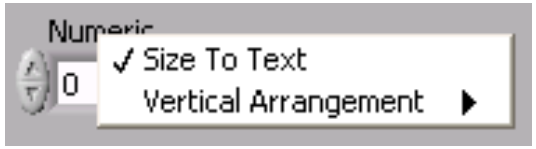
Kontole, indikatori, kontejneri, tab-ovi,...

Podešavanje objekata *Front Panel*-a

- Desni klik za prikaz *shortcut* menija.
- Izborom *Properties* sve karakteristike objekta postaju dostupne







Podešavanje labele kontrole

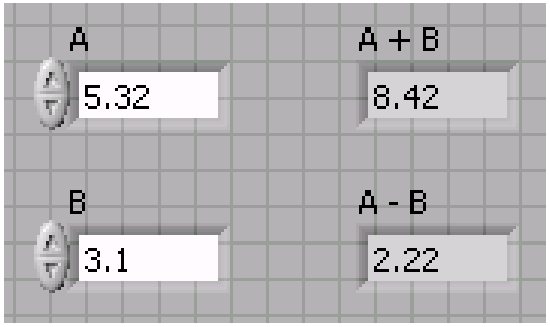
Vežba 1

- Kreirati front panel koji sadrži sledeće kontrole/indikatore:

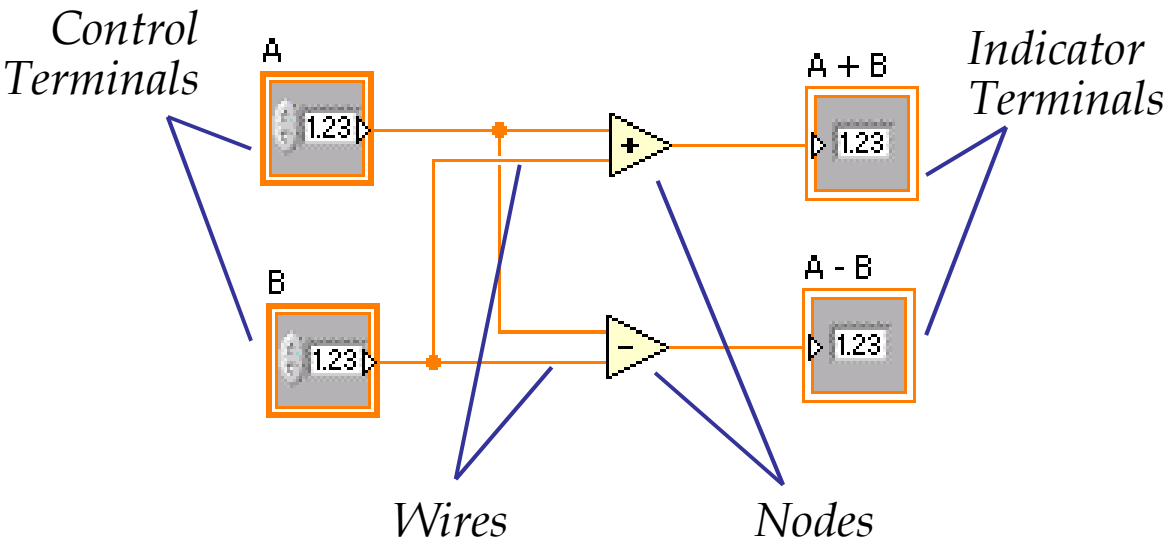
Uloga	Tip podataka	Kontrola/ indikator
Prikaz vrednosti temperature sobe	Numerički	Indikator
Prekid programa		
Username i password		
LED za prikaz greške		

Kreiranje *Block Diagram*-a

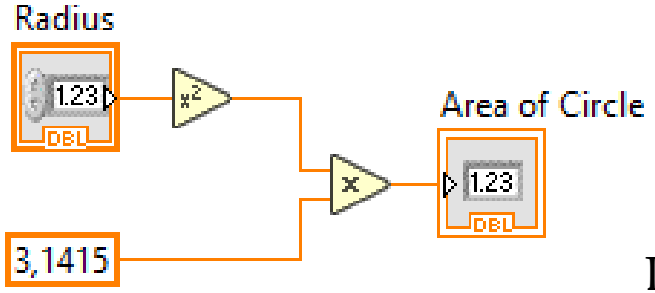
Front Panel



Block Diagram

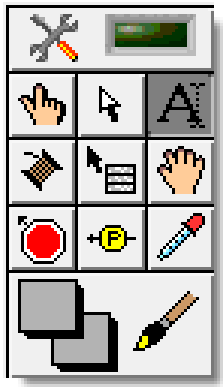


Osim terminala postoje i konstatne.



Dodavanje komentara na *BP*-u.

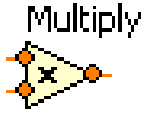
Racuna se povrsina kruga na osnovu zadatog poluprecnika



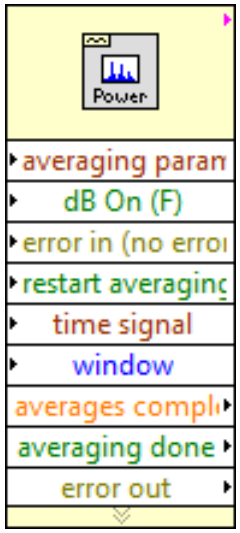
Kreiranje *Block Diagram*-a

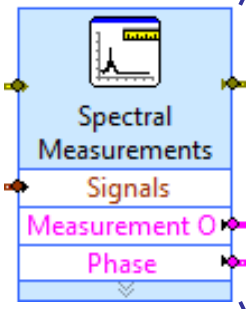
- Čvorovi (*Nodes*) – objekti na *BD* koji imaju ulaze i izlaze i obavljaju operacije kada se VI pokrene:
  - *Functions* (*Function Nodes*): fundamentalni funkcijski blokovi LabVIEW-a, ne sadrže ni *FP* ni *BD*.
  - Standardni VI (koriste se kao *SubVI*): funkcije čiji se parametri rada mogu zadati programski, sadrže i FP i BD.
  - *Express VI*: interaktivni VI koji poseduju *dialog* za konfigurisanje.

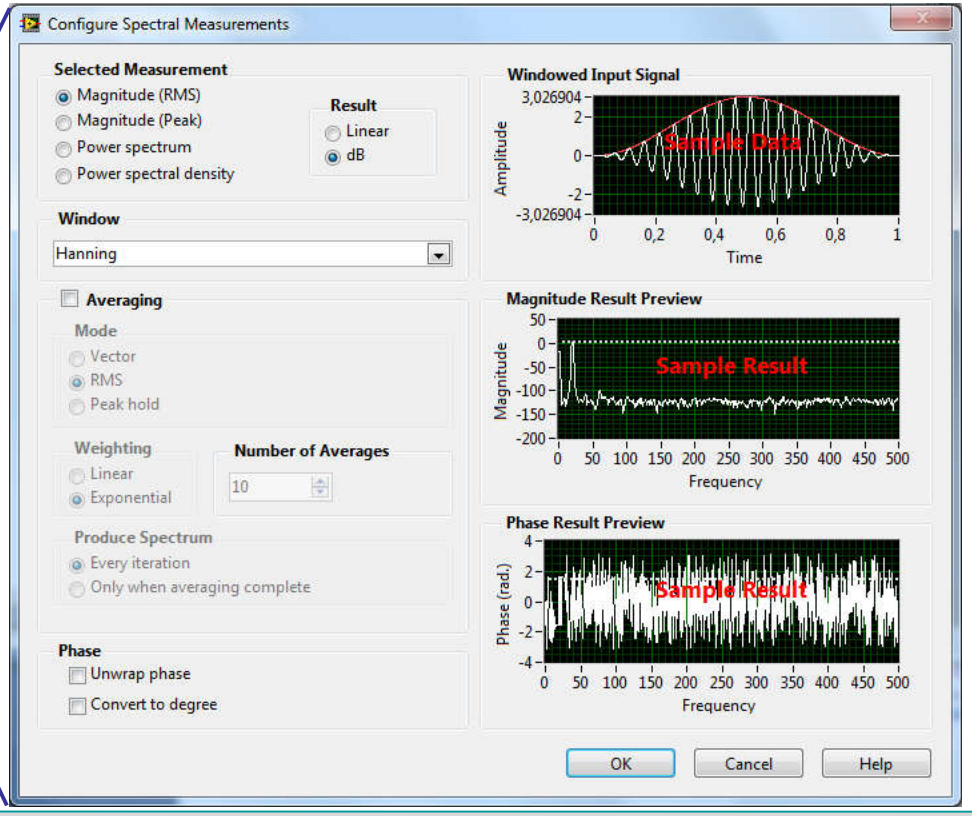
Functions



Expanded VI

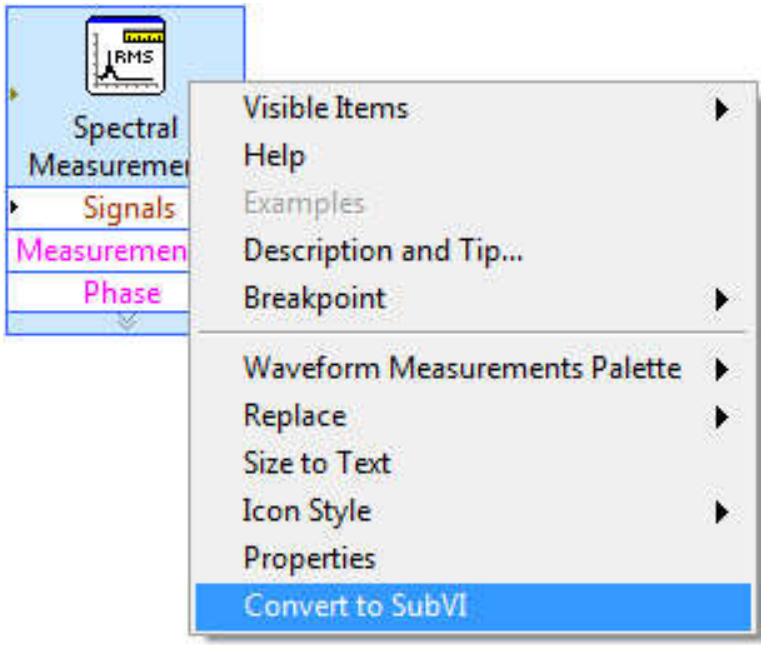


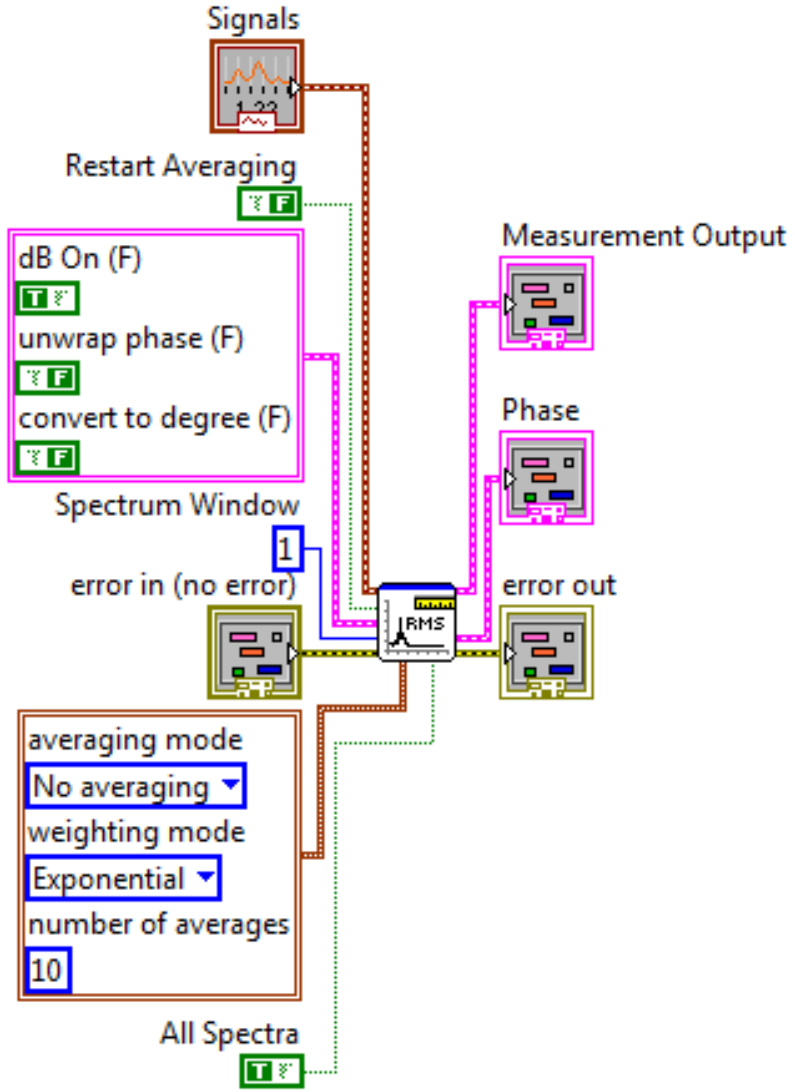




Kreiranje *Block Diagram*-a

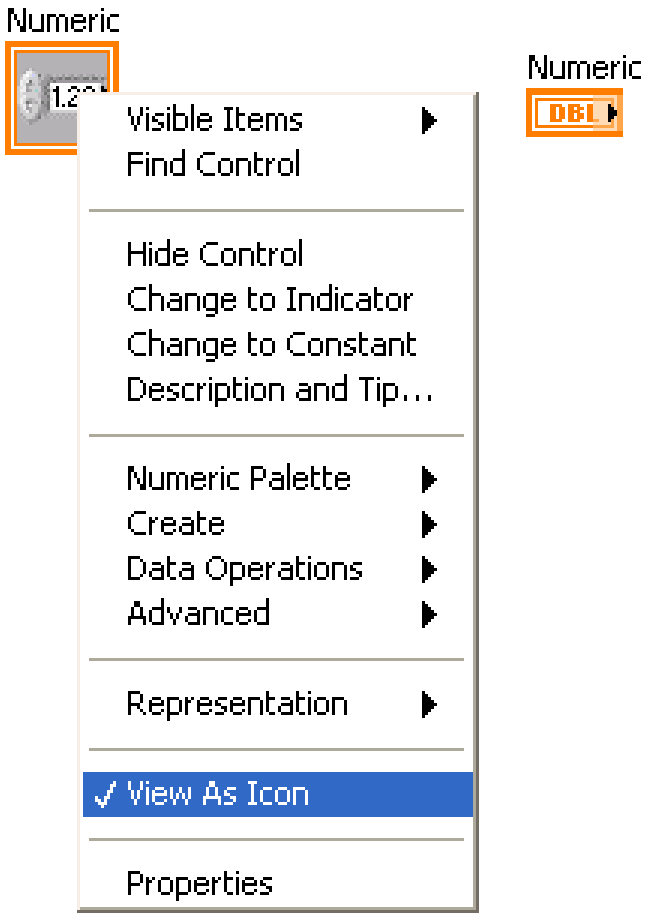
- *Express VI* – može poslužiti kao osnova pri razvoju aplikacije, a potom je moguće preuzeti samo deo koda i iskoristiti ga za finalnu aplikaciju.





Kreiranje *Block Diagram*-a

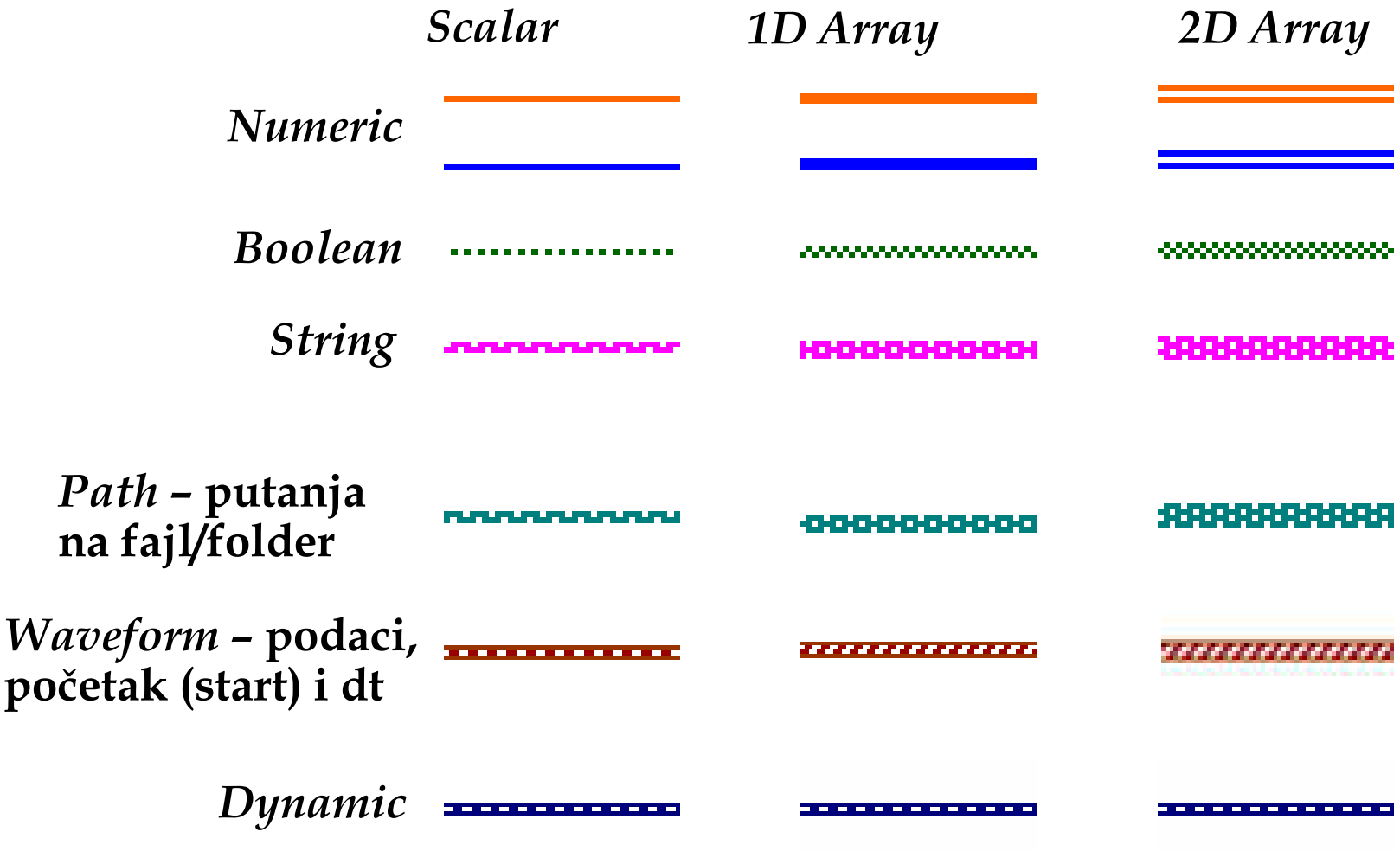
- Terminali – ulazi i izlazi *BD*-a, koji služe za razmenu informacija između *FP*-a i *BD*-a.
- Terminali su analogni ulazno izlazni parametrima u tekstualnom programiranju.
- Desni klik i promena vrednosti polja *View As Icon* menja način prikazivanja terminala.



Kreiranje *Block Diagram*-a

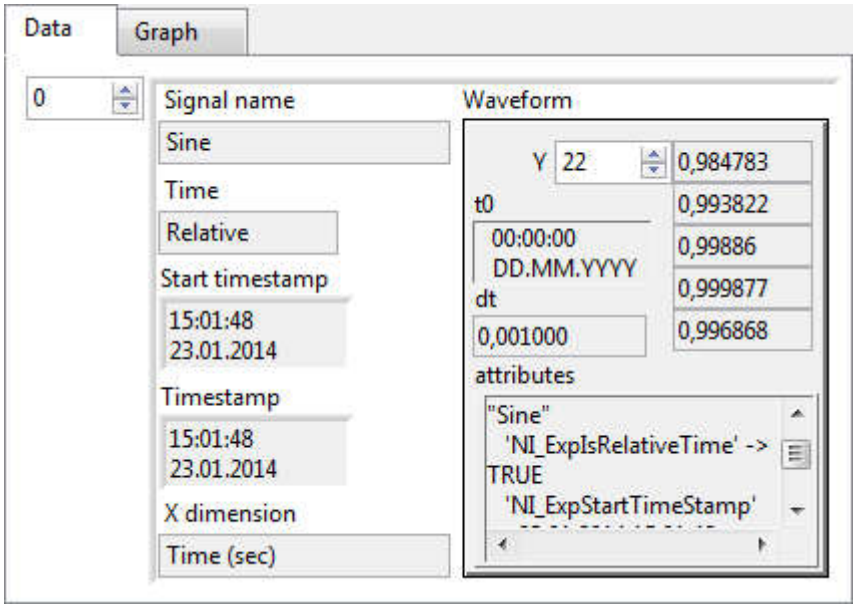
Ožičavanje *Block Diagram*-a:

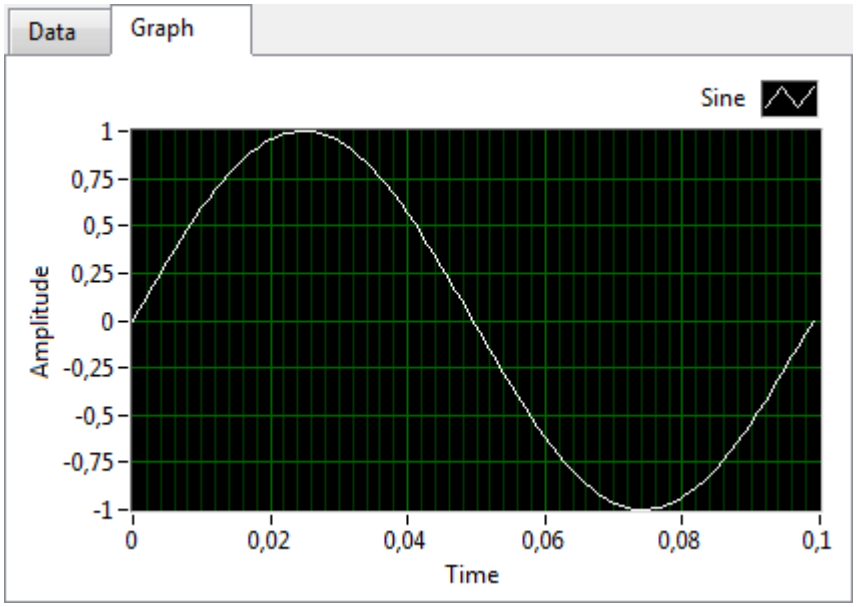
Svakom osnovnom tipu podataka odgovara drugačija boja žica (veza).
Numeric može biti celobrojni (plavo) ili realni broj narandžasto



Kreiranje *Block Diagram*-a – *Dynamic Data Type*

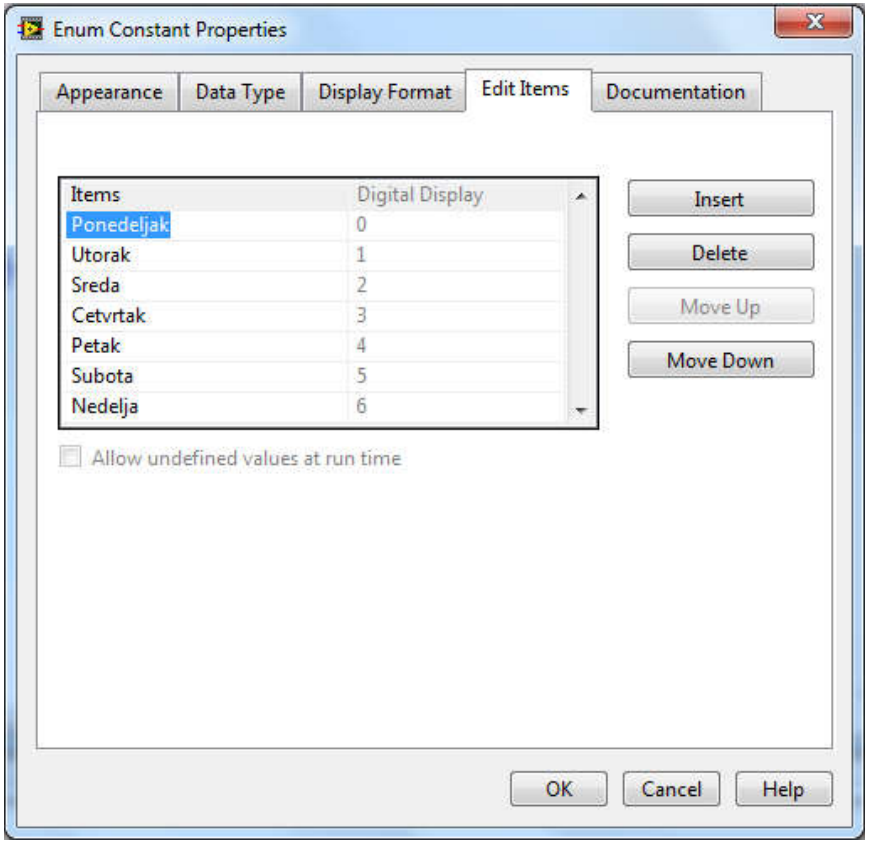
- *Dynamic Data Type* – tip podataka koji se dodeljuje *Express VI* za akviziciju, obradu ili generisanje signala (*waveform*).
- Osim vrednosti odbiraka (niz) *Dynamic Data Type* sadrži dodatne podatke – kada je izvršena operacija na signalu, kada je signala generisan, koliko je vremenski razmak između odbiraka, itd.

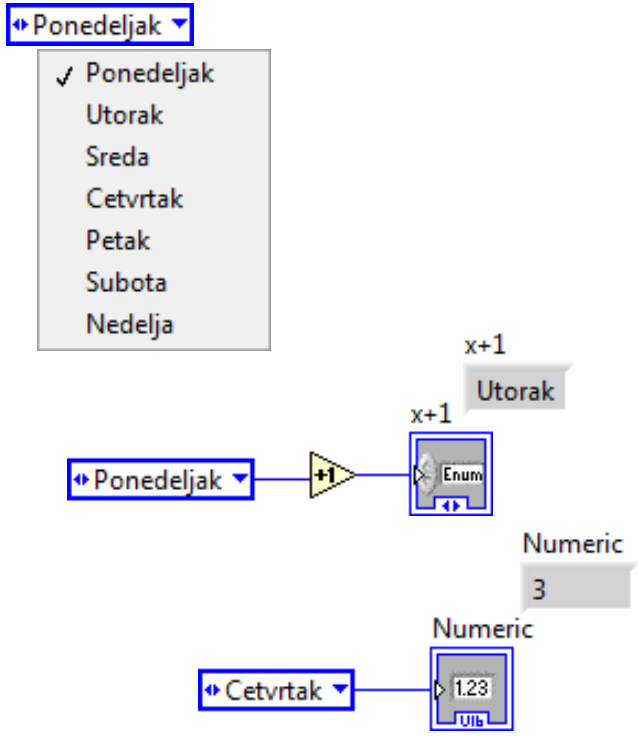




Kreiranje *Block Diagram*-a – *Enum* tip podatak

- *Enum* pretstavlja skup elemenata.
- Svakom elementu se može dodeliti proizvoljno ime.
- Svaki element se pretsavlja brojem tipa U16, i sve operacije koje važe za U16 važe i za *Enum*.
- Kao terminal pogodnije je koristi *Enum* indikatore ili kontrole, jer se tada tačno može videti ime elementa.

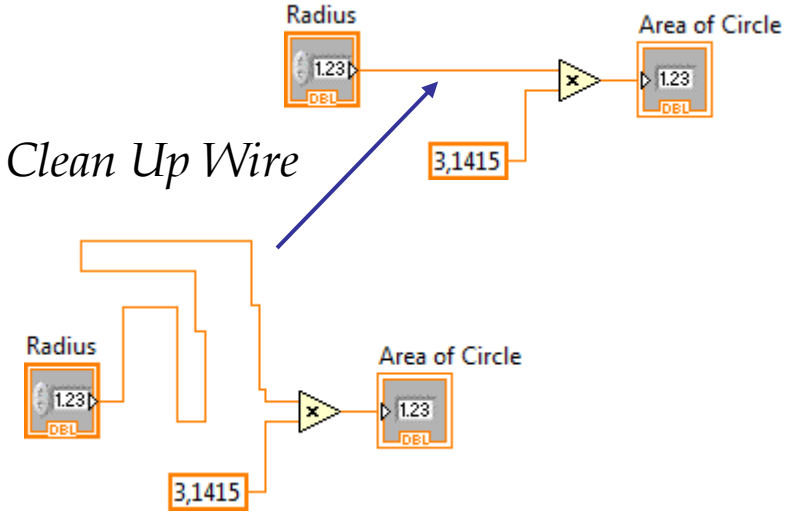


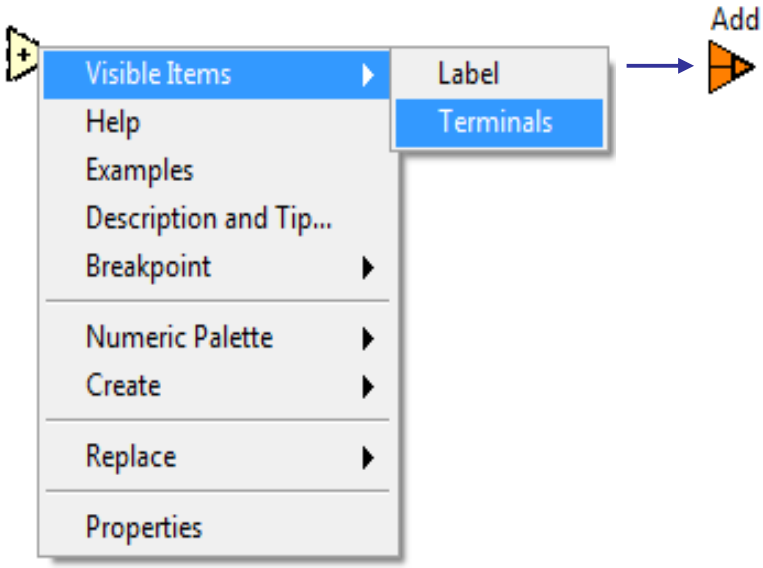


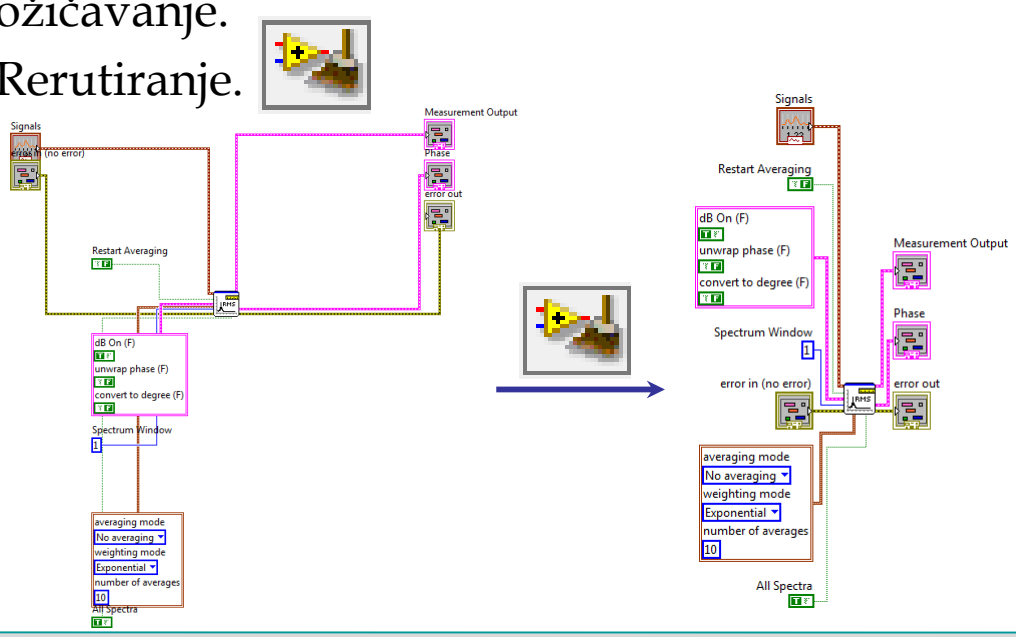
Kreiranje *Block Diagram*-a

Hot Spot → Alat za ožičavanje

- Poželjno je koristi *Context Help Window* pri ožičavanju terminala/konstanti i čvorova.
- Desni klik na žicu i izborom ***Clean Up Wire,*** omogućava se preglednije povezivanje elemenata *BD*-a.
- Desni klik na čvorove ***Visible Items*** » ***Terminals*** omogućava jednostavnije ožičavanje.
- Rerutiranje.



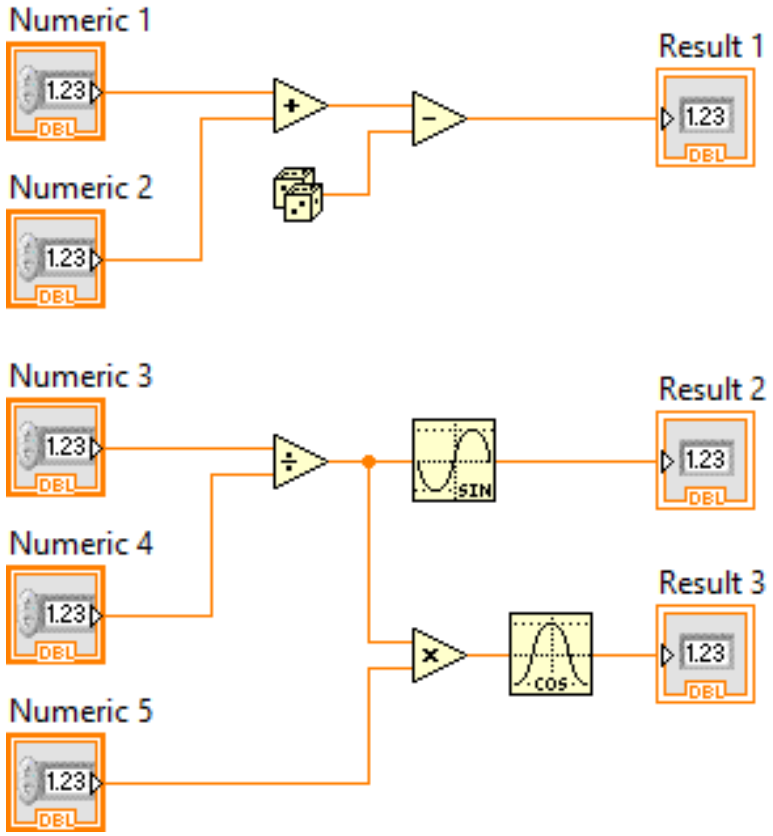




Vežba 2

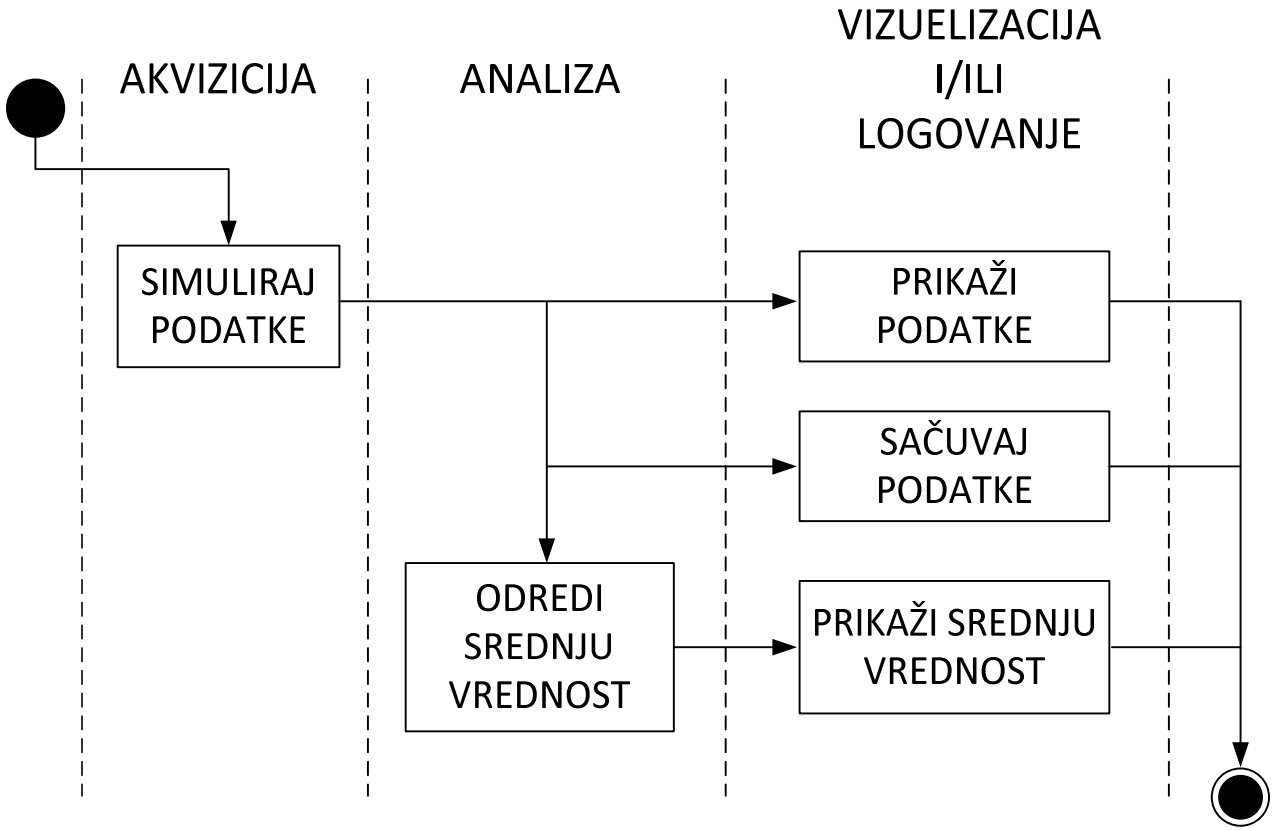
- Pronaći VI “Format Date/Time String“ dodati ga na *Block Diagram*, a zatim na osnovu dokumentacije podesiti “Format Date/Time String“ tako da se dobije poruka u sledećem formatu “Trenutno vreme i datum su: 10:15:45 i 26.10.2018“

- *BD* se izvršava kako podaci postaju dostupni, a ne sa leva u desno.
- Čvor se izvršava tek kada **SVI** podaci postanu dostupni.
- Kada se čvor izvrši podaci postaju dostupni na izlazu.
- Za *BD* prikaza na slici nemoguće je utvrditi koji se čvor izvršava prvi. Jedino se može tvrditi da se operacija MINUS izvršava nakon PLUS i RANDOM NUMBER, kao i da se funkcija SINUS izvršava nakon operacije deljenja.
- Takođe, nije sigurno da će se funkcija SINUS izvršava pre funkcije COSINUS.



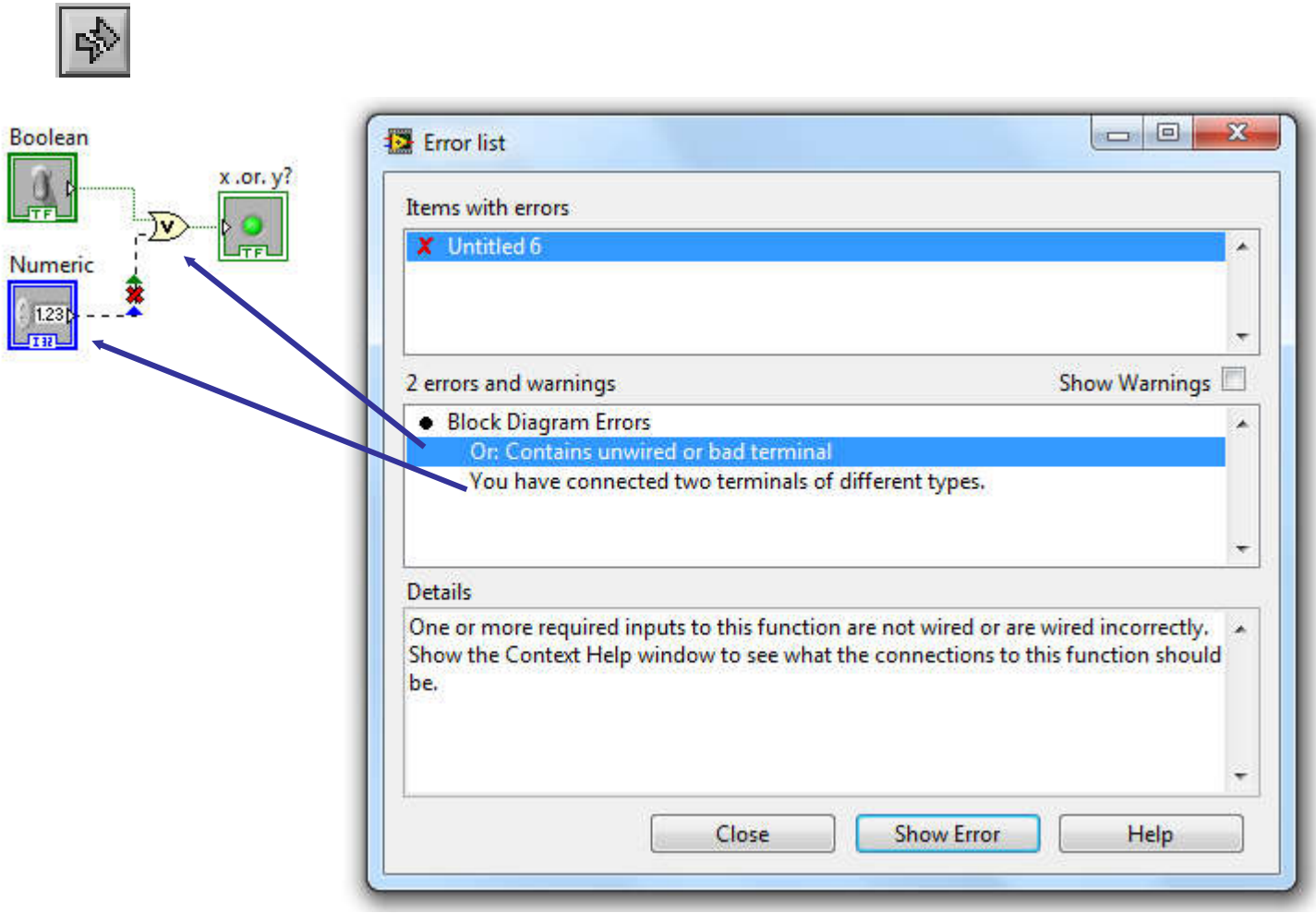
Vežbe 3 i 4

- Vežba 3: Napraviti VI koji određuje površinu pravougaonika na osnovu dužine uneti stranica: $A=\text{sqrt}(s(s-a)(s-b)(s-c))$, gde je $s = (a+b+c)/2$.
- Vežba 4: Napraviti VI koji koristi sledeće Express VI: *Simulate Signal*, *Statistics* (*find Avg Value*), *Write to Measurement File* i dodati *Waveform Graph* za prikaz rezultata.



Debugging Tehnike

- **Pronalaženje grešaka** – najčešći uzroci – spajanje različitih tipova podataka, nisu svi ulazi *function node*-ova spojeni, nisu definisani *required* ulazi *SubVI*-a.
- Izbor *Run button* sa greškom. Pojavljuje se prozor sa greškama (*Error list*).



Debugging Tehnike

Execution Highlighting

Usporen animiran prikaz toka podataka, pri čemu su vrednosti podataka date na žicama.

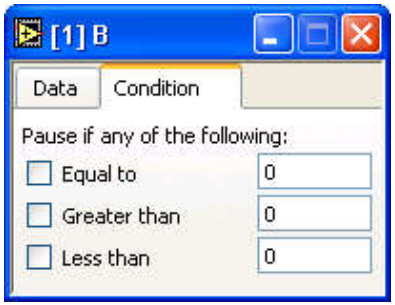
Probe

Desni klik na žicu i izbor *Probe* omogućava prikaz vrednosti podataka koji prolazi kroz posmatranu žicu.

Breakpoints

Desni klik na žicu i izbor *Set Breakpoint* obezbeđuje pauziranje izvršavanja *BD*-a kada podaci stignu do *Breakpoint*-a.

Conditional Probe



Kombinacija *Breakpoint*-a i *Probe*. Desni klik na žicu izbor *Custom Probe* » *Conditional Boolean Probe*, a zatim se definiše uslov pauziranja na osnovu vrednosti podatka koji prolazi kroz tačku *Breakpoint*-a.

Debugging Tehnike

Step Into

Pokreće/nastavlja izvršavanja programa korak po korak. Omogućava i ulazak u *SubVI*-a (potprogram) i izvrašavanje *SubVI*-a korak po korak.

Step Over

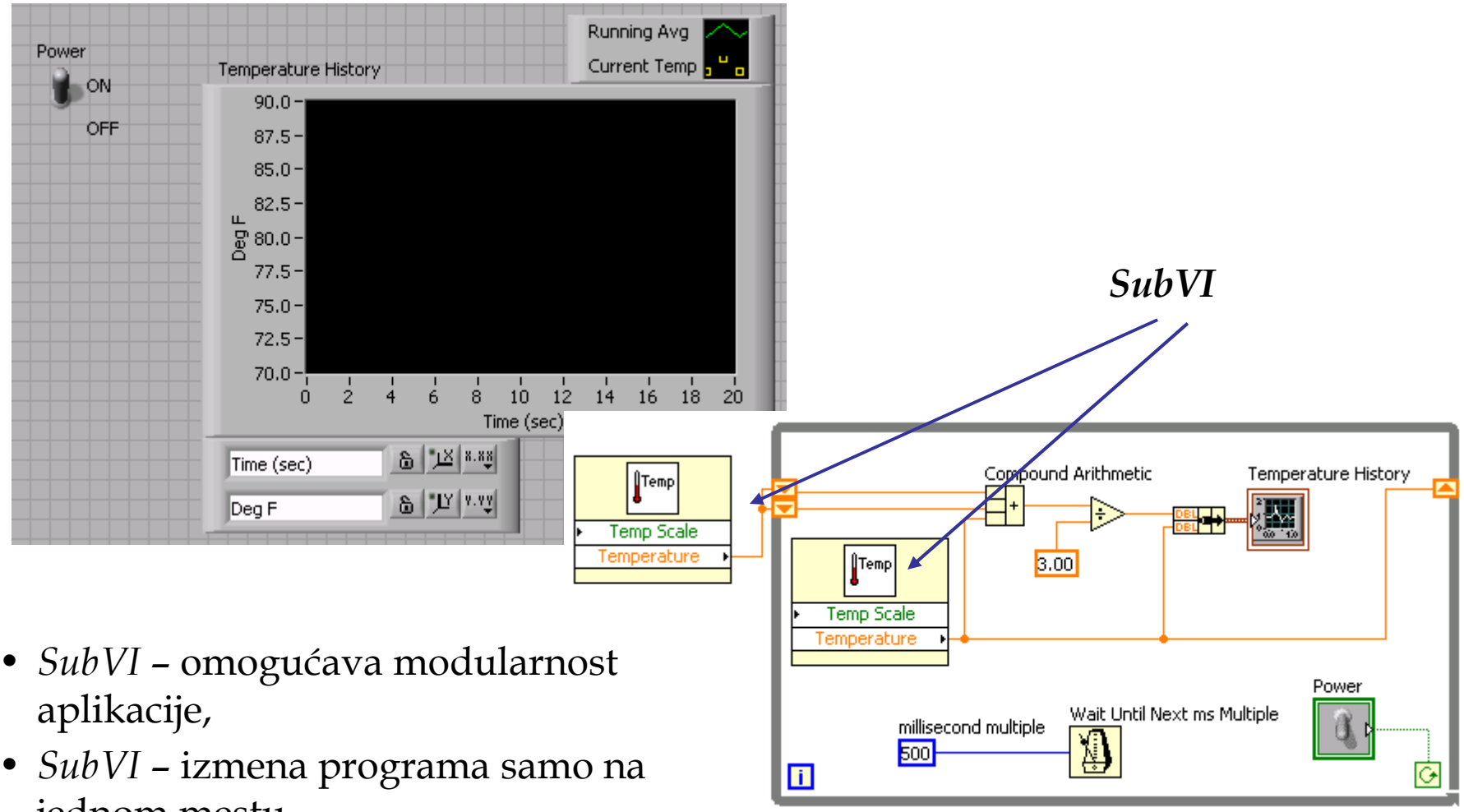
Ako je je pokrenuto *Step Into* izvršavanje, *Step Over* obezbeđuje da se *SubVI* posmatara kao funkcija, tj. *subVI* se izršava u jednom korak.

Step Out

Aktiviranjem *Step Out* završava se korak po korak izršavanje potpograma i i nastavlje se sa *Step Into* odakle je pozvan potrogram ili se prelazi se u standardan režim izvršavanja programa ako je *Step Out* aktiviran iz glavnog programa.

Vežba 5: testirati *Execution Highlightning* i *Sinle Stepping* na VI koji računa površinu trougla (vežba 3).

Hijerarhija LabVIEW-a



- *SubVI* – omogućava modularnost aplikacije,
- *SubVI* – izmena programa samo na jednom mestu,
- *SubVI* – neograničen broj ugnježdavanja.

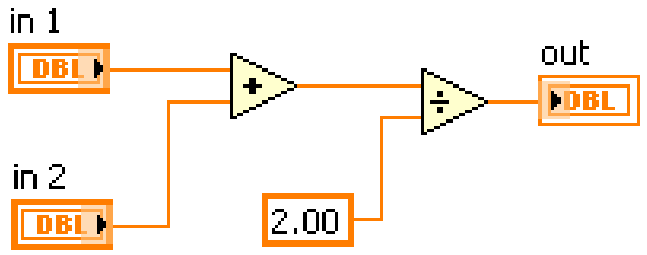
Function Pseudo Code

```
function average (in1, in2, out)
{
out = (in1 + in2)/2.0;
}
```

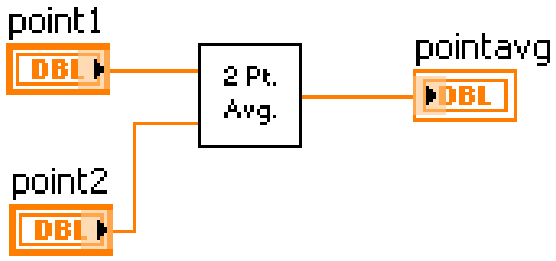
Calling Program Pseudo Code

```
main
{
average (point1, point2, pointavg)
}
```

SubVI Block Diagram

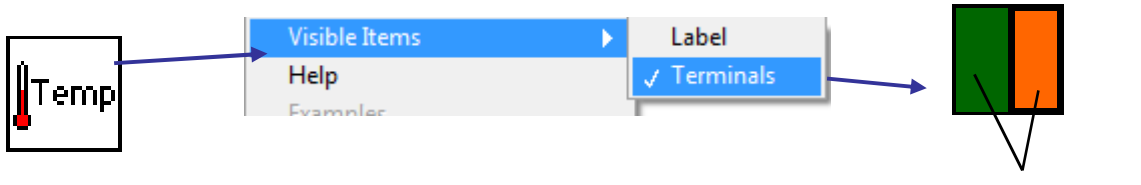


Calling VI Block Diagram

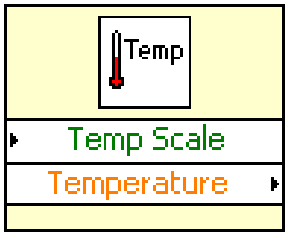


Icon/Connector

Ikonom se predstavlja *subVI* na *BD*-u. Preko terminala se prosleđuje i prihvataju podaci.

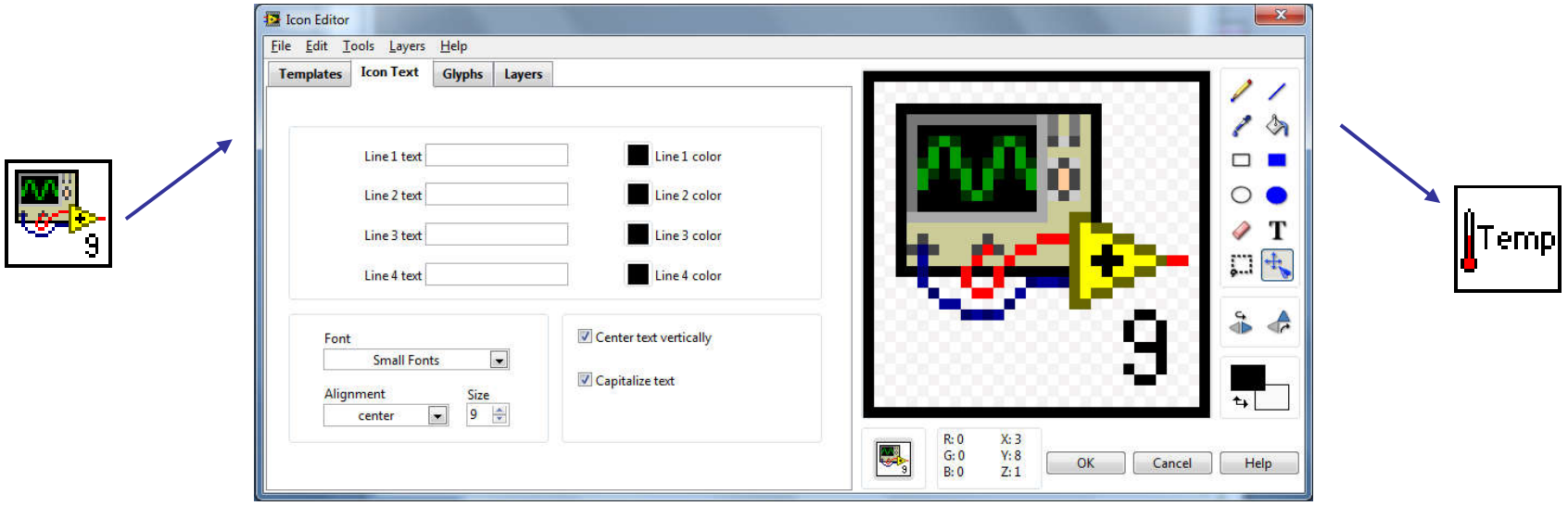


terminali



terminali

Moguć i objedinjen prikaz ikone i terminala.



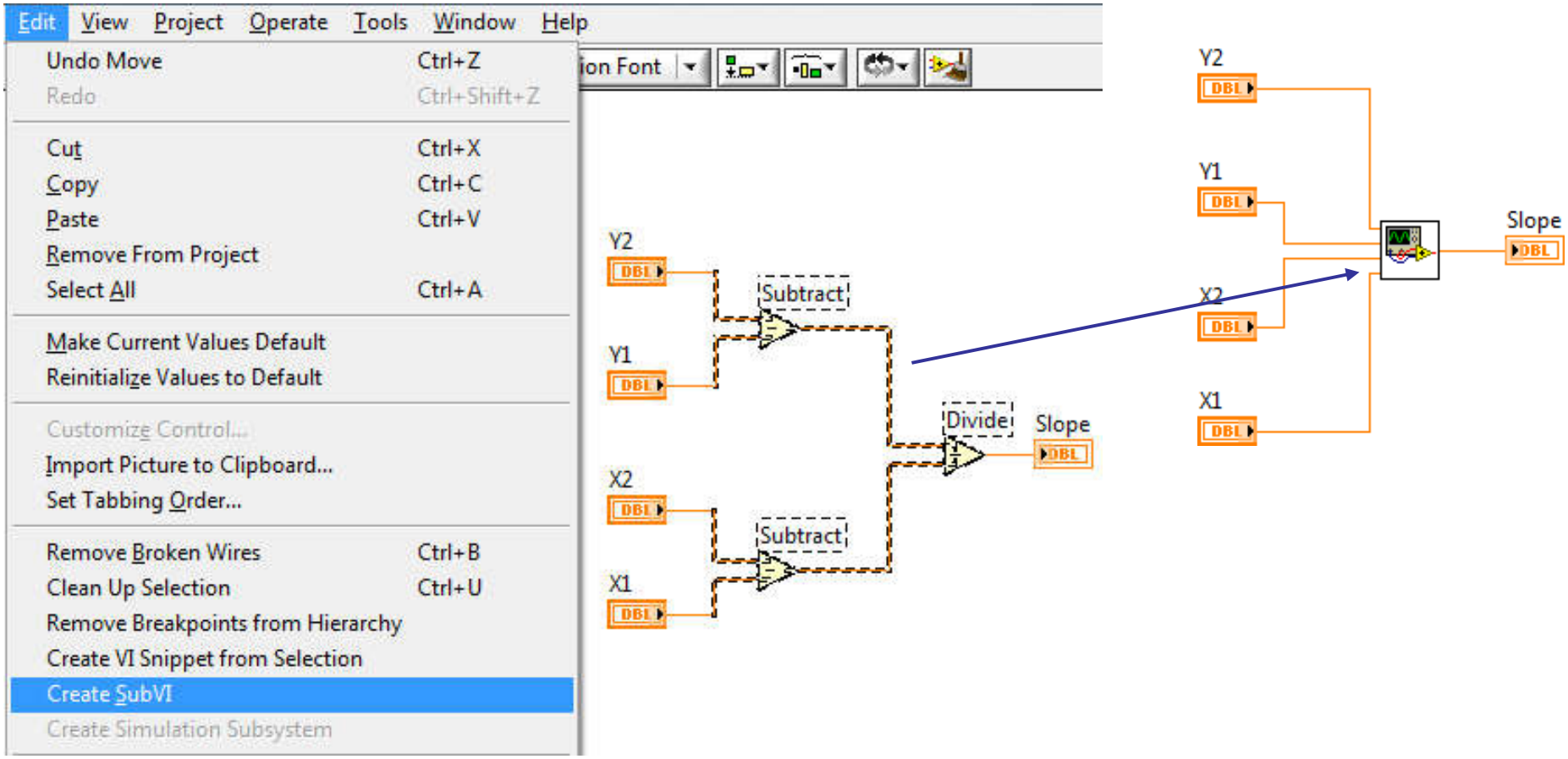
Ikona je grafički prikaz *subVI*-a.

Omogućava lakše razumevanje BD-a.

SubVI – kreiranje na osnovu dela *BD*-a glavnog *VI*-a.

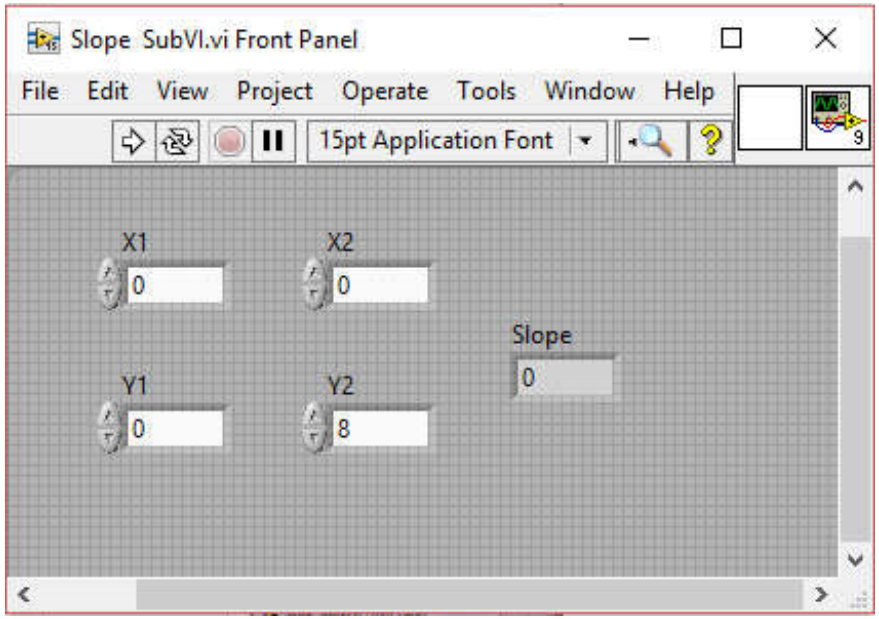
Deo koda BD-a može se zameniti SubVI.

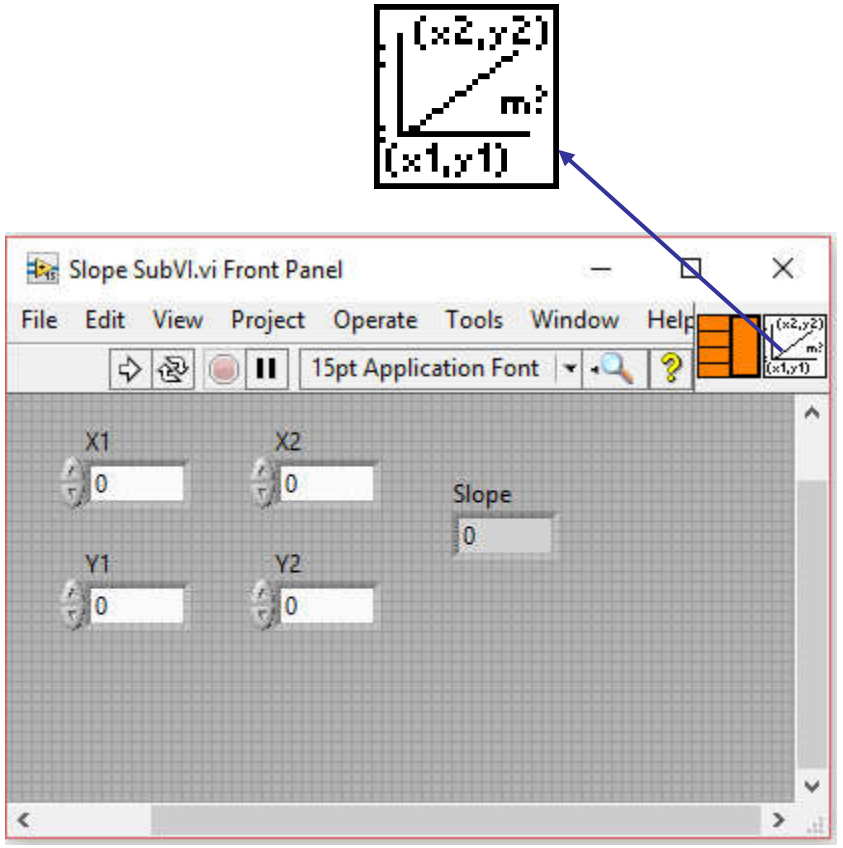
Selektovati deo koda a zatim *Edit* » *Create SubVI.*



SubVI – Dizajn ikone

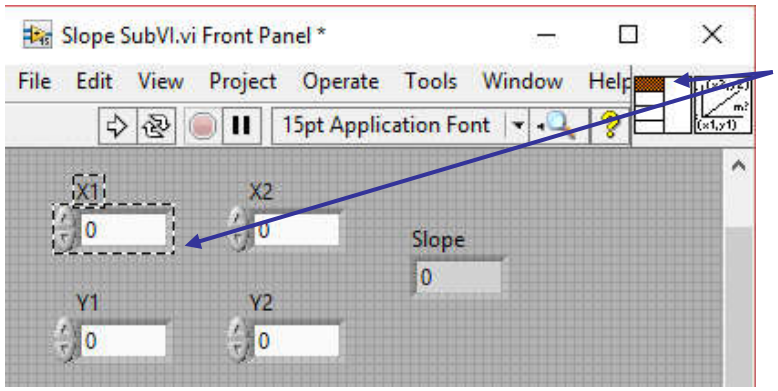
Ikona grafički predstavlja *SubVI* na *BD* glavnog programa (*VI*-a).
LabVIEW zadaje standardnu ikonu svakom *VI*.
Moguće je da sam korisnik dizajnira ikonu – dvoklik na ikonu.

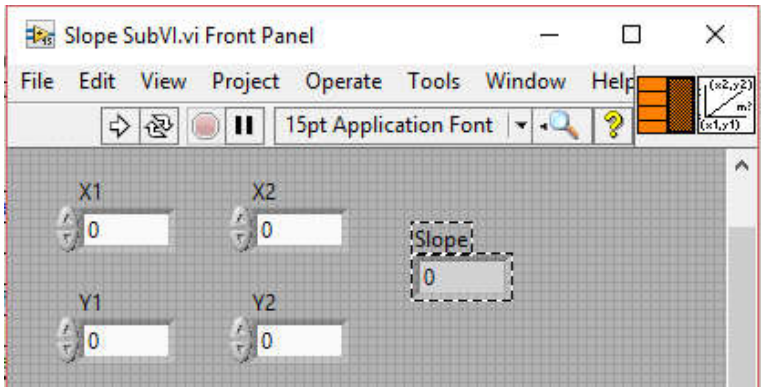


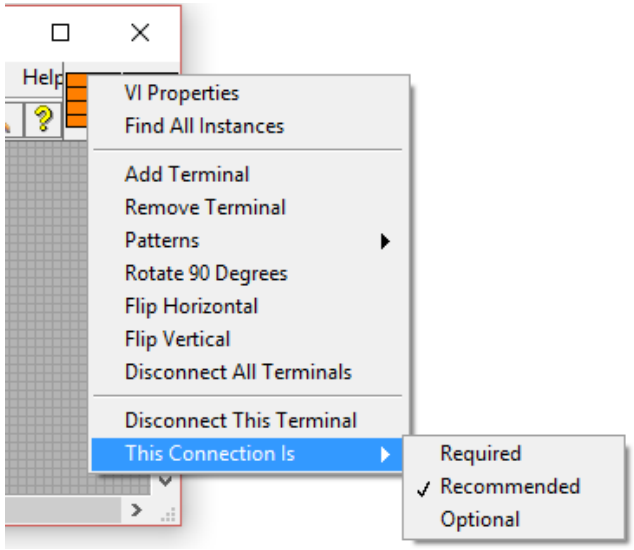


SubVI – Definisanje *Connector*-a (*Connector Pane*)

- Definisanje ulazni i izlaznih parametar SubVI-a.
- *Wiring* alatom povezuju se terminala konektora sa terminalima *FP*-a.
- Veliki izbor šablona konektora.
- Prelaskom miša preko terminala konektora na FP-a se označava odgovarajući terminal.



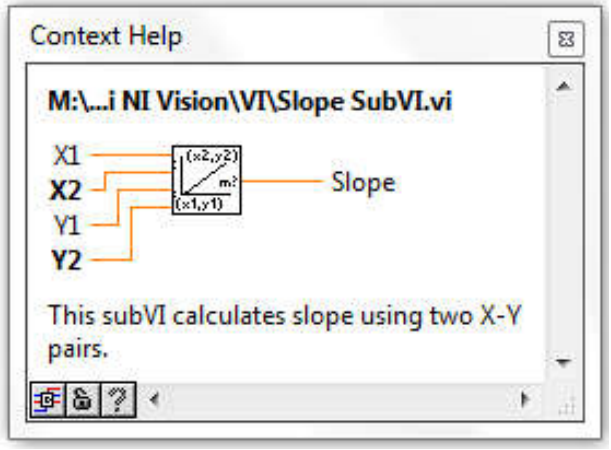




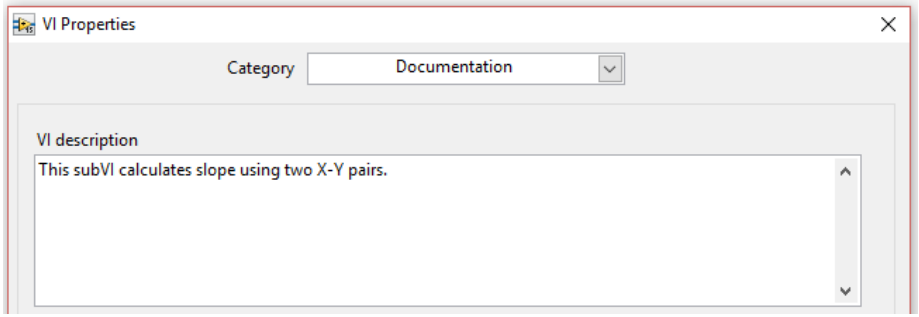
Klasifikacija terminala konektora:

- *Required* (*bold*) – Greška kada se ne poveže.
- *Recommended* – Upozorenje ako se ne poveže
- *Optional* – bez uticaja.

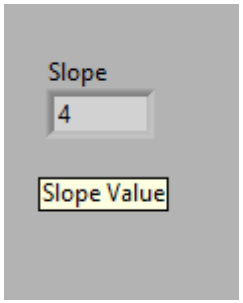
SubVI – Definisanje *Connector*-a i dokumentacije

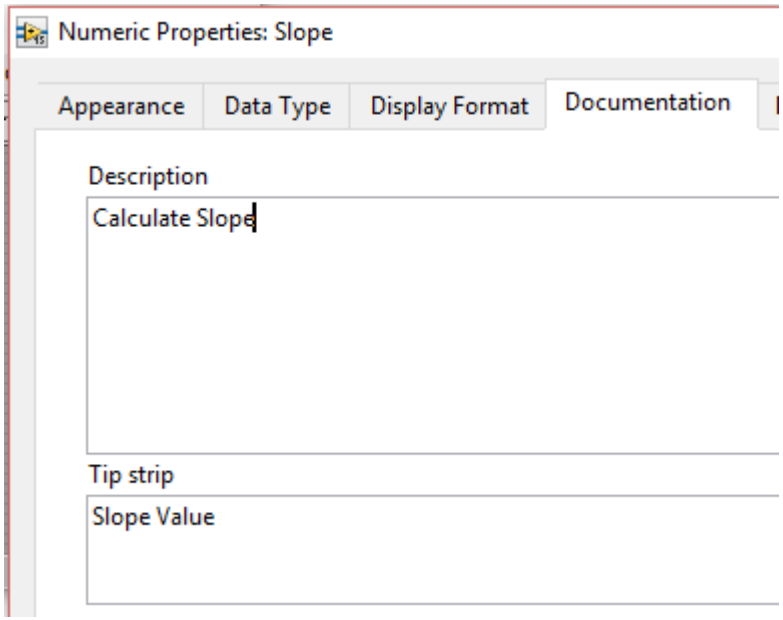


Definisanje dokumentacije – tekst kojim se opisuje *SubVI* u *Context Help*-u:
File » VI *Properties* » *Documentation*.



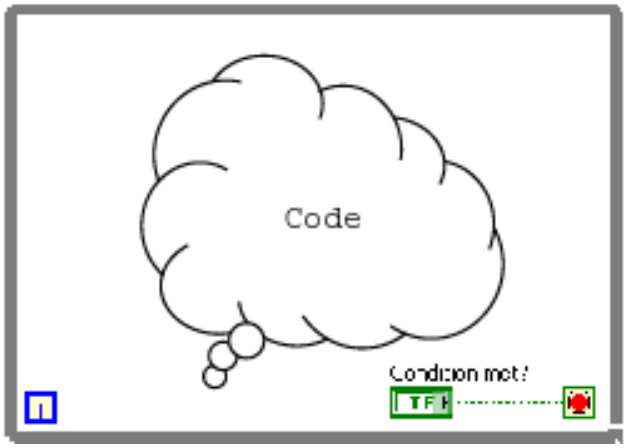
Za objekat (kontrola ili indkator) na FP-u, osim dokumentacije može se definisati i *Tip strip*, koji se ispisuje kada se kursorom pređe preko objekta u toku izvršavanja programa.



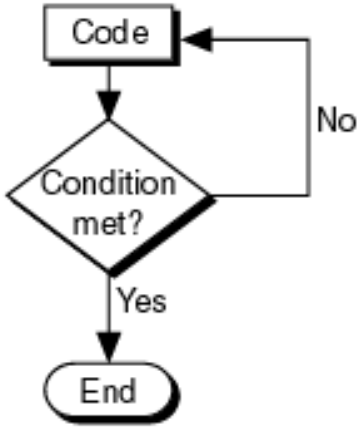


While loop

Kod unutar *While* petlje izvršava se bar jednom.



LabVIEW *While Loop*

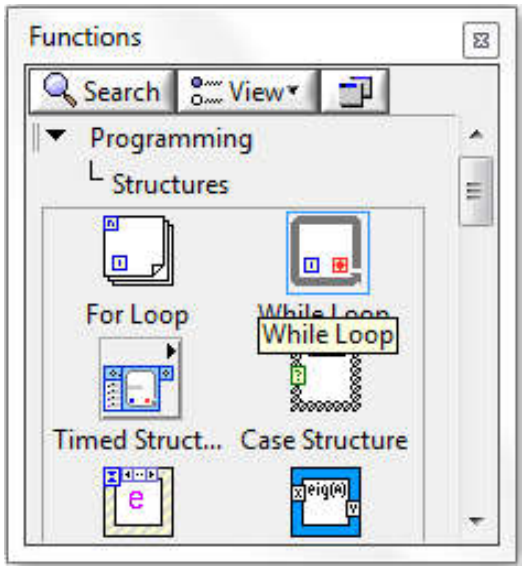


Flow Chart

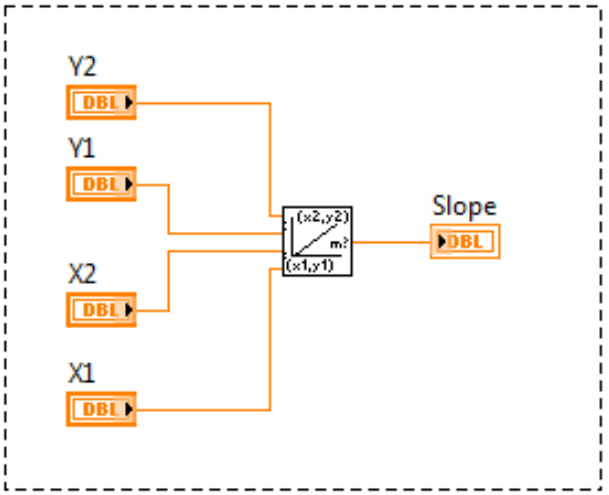
```
Repeat (code);
Until Condition met;
End;
```

Pseudo Code

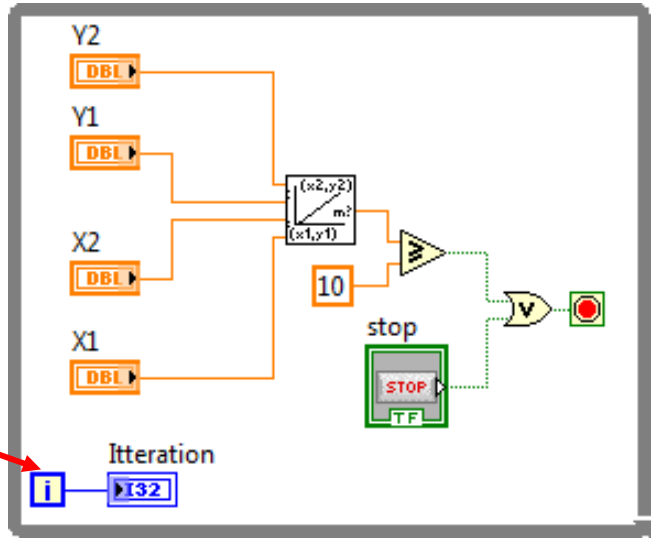
While loop



Izabrati *While loop* sa palete funkcija.



Označiti deo koda koji je potreno izvršiti više puta.

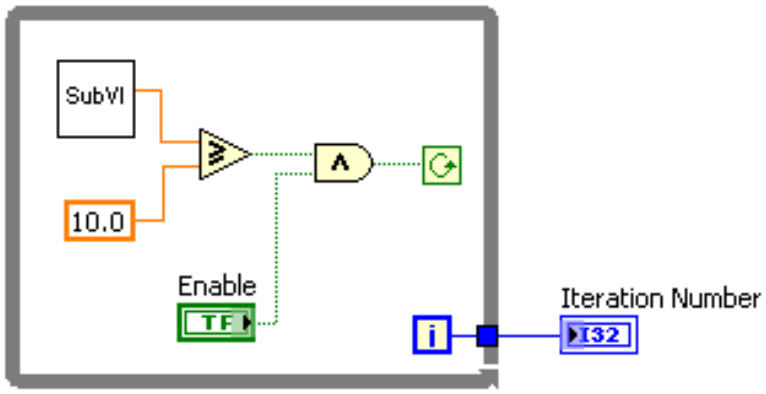


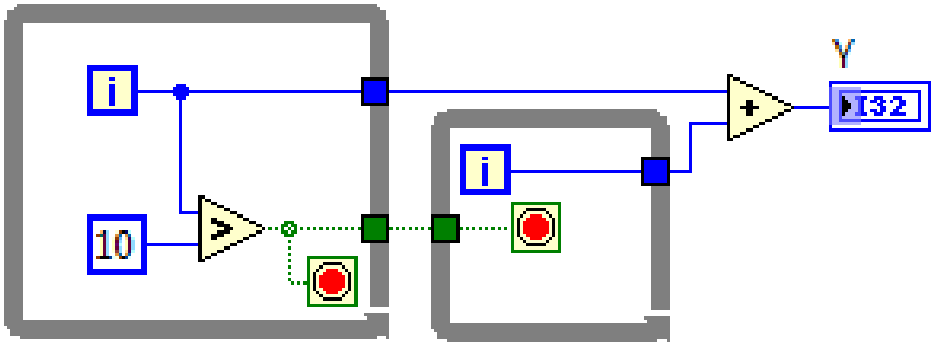
Broj trenutne iteracije.

Kreirati dodatni kod unuta petlje.

Tuneli u strukturama

- Tuneli (*Tunnels*) uvoze i izvoze podatke u stukture (*while loop*, *for loop*, *case structure*, *sequence*,...)
- Tunel je kvarat koji se pojavljuje na ivici struktre, a boja tunela odgovara boji tipa podatka koji prolazi kroz tunel.
- Kada tunel prosleđuje podatke petlji, petlja počinje sa izvršavanjem tek kada se podatak stigne do tunela.
- Podatak je dostupan na izlaznom tunelu tek kada se završi izvršavanje petlje.

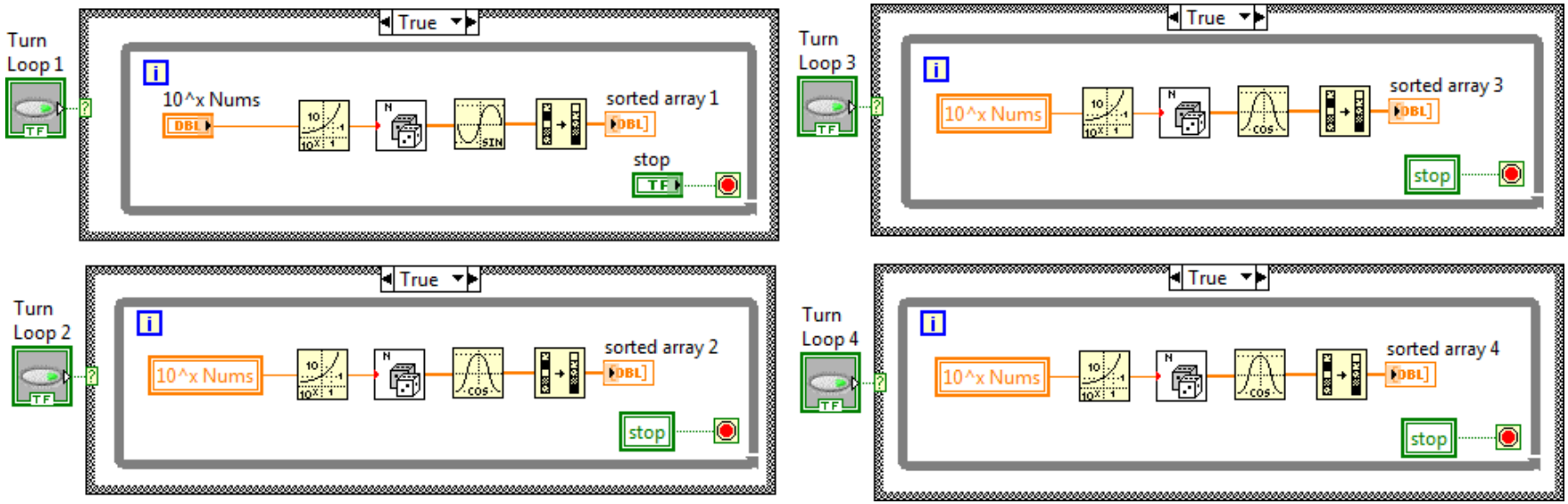




Koliko je Y?
11, desna petla
se izvrši samo 1.

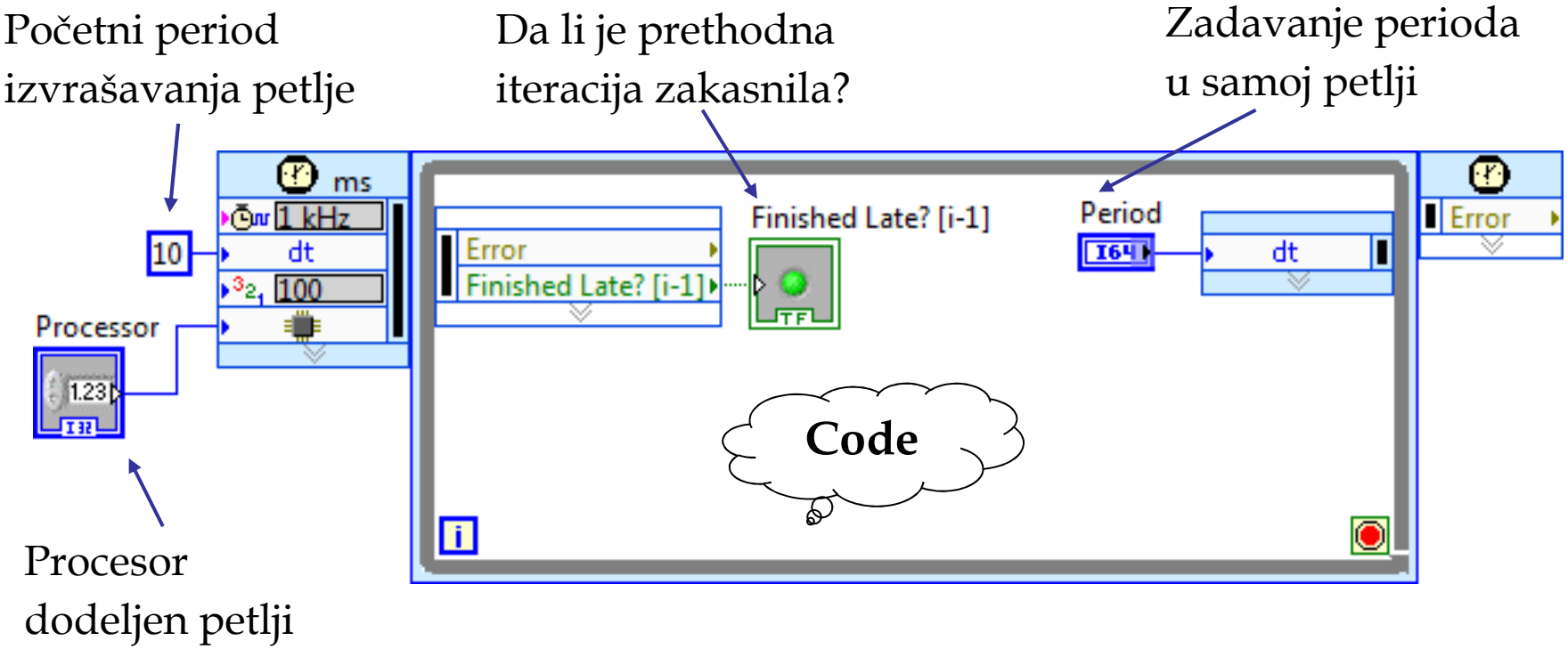
While loop – paralelno izvršavanje

- U zavisnosti od broja jezgara LabVIEW automatski vrši *multitasking*.
- D*ata-flow* model programiranja je po svojoj prirodi paralelan i svaki nezavisan *data-flow* deo koda se može paralelno izršavati sa drugim nezavisnim *data-flow* delom koda.



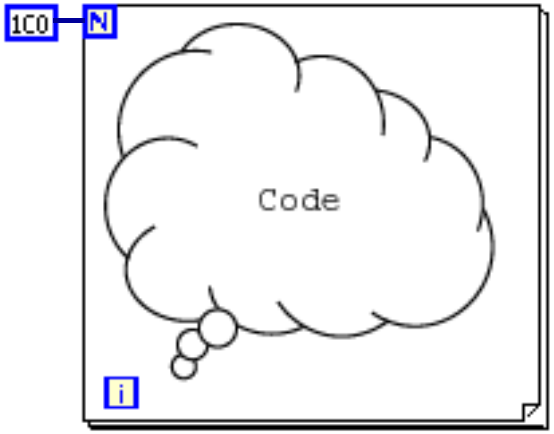
Timed while loop

- Omogućava, ukoliko hardverski resursi to dozvaljavaju, izvršavanje koda petlje sa tačno zadatim periodom, koji se može programski menjati u petlji.
- Period petlje se može zadati sa korakom od 1 ms ako ne postoji drugi generator takta, a moguć je korak i od 1 μs ukoliko postoji odgovarajući hardver.
- Program može prepusti LabVIEW-u da automatski dodeli procesor na kome će se izvršavati petlja (-2), a može i sam program to da definiše (0 do br. procesora – 1).
- Moguća sinhronizacija više *time while loop*.

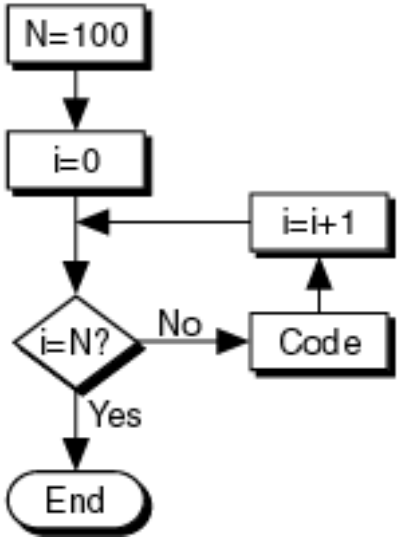


For loop

Kod unutar *For* petlje može se i ne izvršiti.



LabVIEW *For Loop*



Flow Chart

N=100;

i=0;

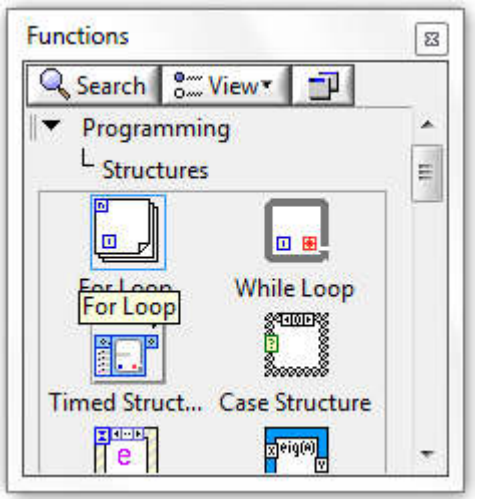
Until i=N:

Repeat (code; i=i+1);

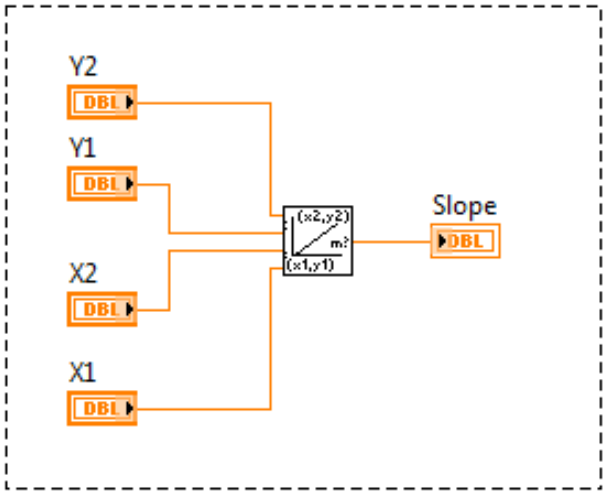
End;

Pseudo Code

For loop

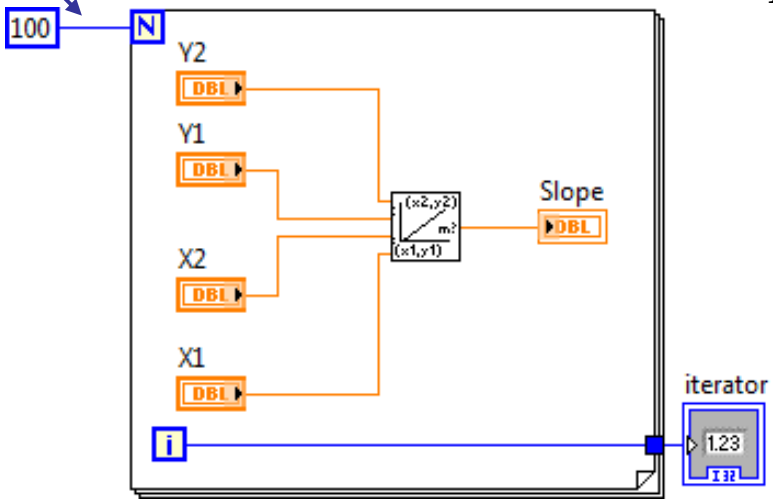


Izabrati *For loop* sa palete funkcija.



Označiti deo koda koji je potreno izvršiti više puta.

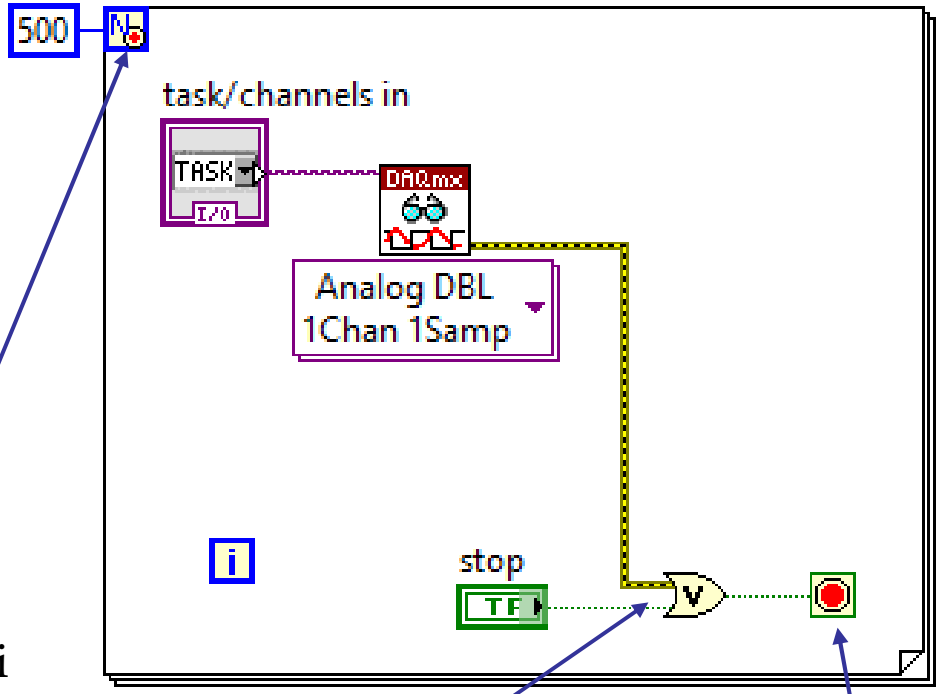
Zadati koliko puta treba petlja da se izvrši (*count terminal*)



?
99, jer je to vrednost zadnje iteracije (***i*** ide od 0 do 99)

For loop

Dodavanje uslovnog terminala (*Conditional Termial*), kojim se može prekinuti rad *for* petlje i pre nego što se izvrši zadat broj iteracija. Desni klik na *for* petlju.



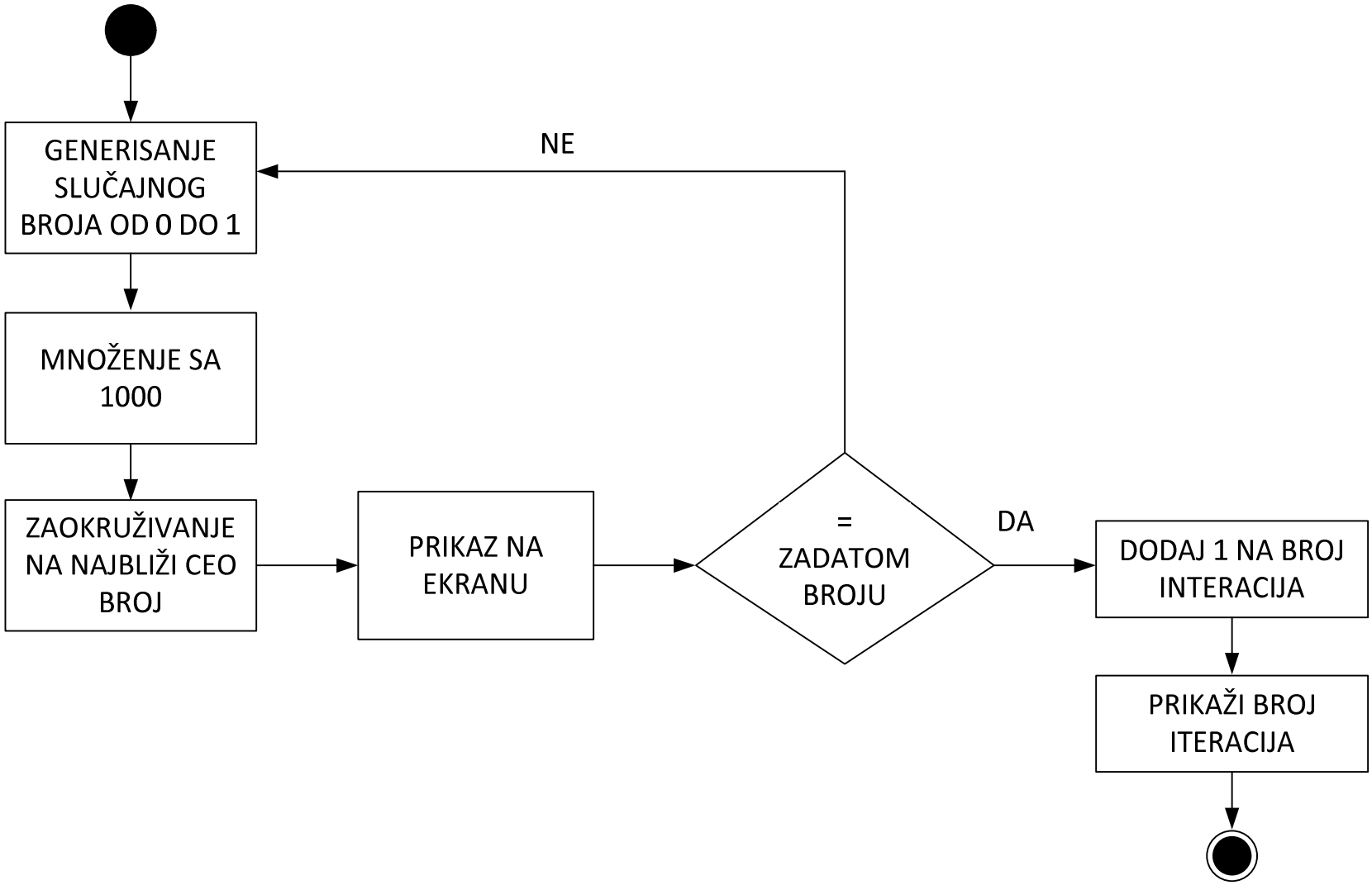
Označava da postoji *Conditional Terminal*

Conditional Terminal

Spajanje dva različita tipa podataka: signal greške (*Error Cluster*) i *boolean*. LV ne prijavljuje grešku u ovom slučaju, jer podrazumeva da se koristi *status* informacija iz signala greške.

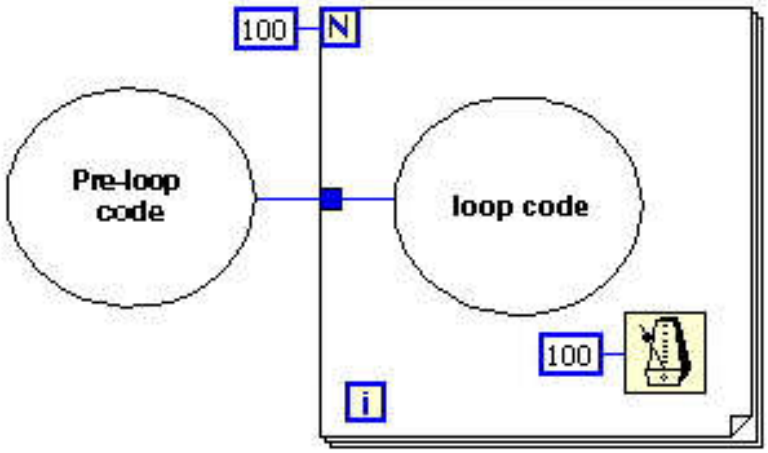
Vežba 6

Napraviti program koji realizuje sledeći dijagram toka:

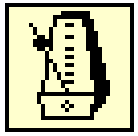


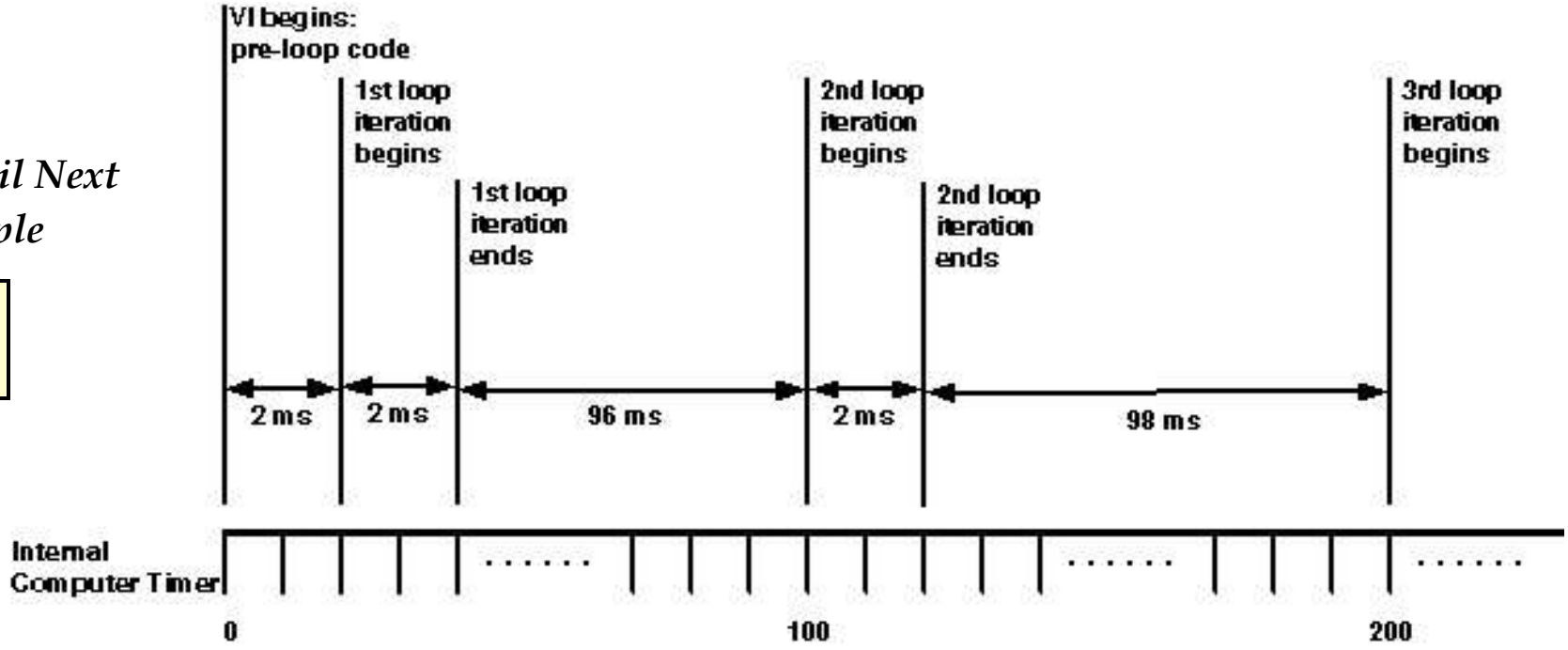
Funkcije za sinhornizaciju

Obezbeđuju da procesor ne bude 100% opterećen ukoliko nema drugih izvora čekanja, npr. akvizicija



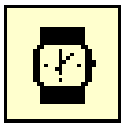
Wait Until Next ms Multiple

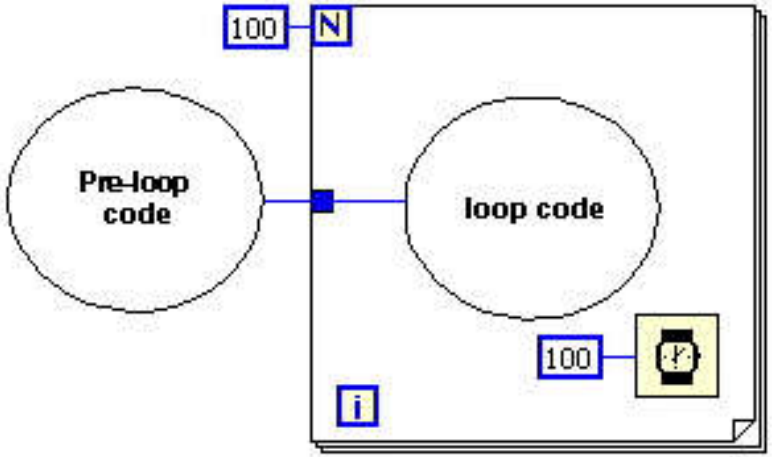




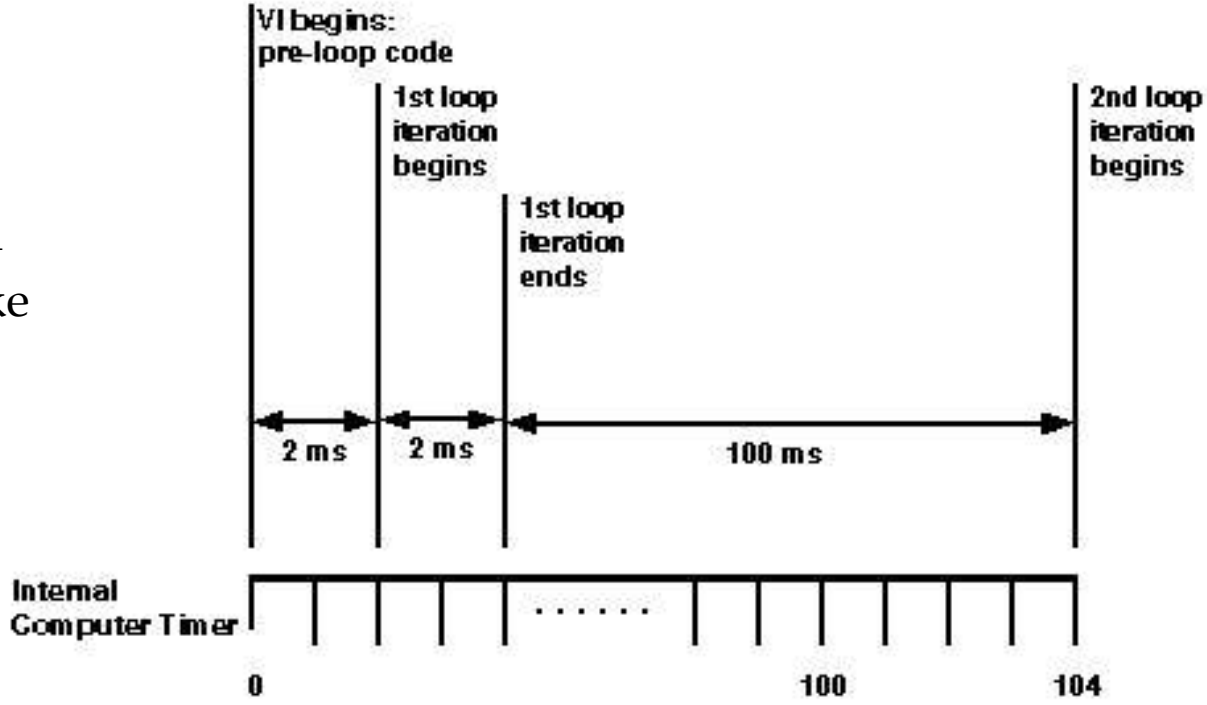
Funkcije za sinhornizaciju

Wait (ms)

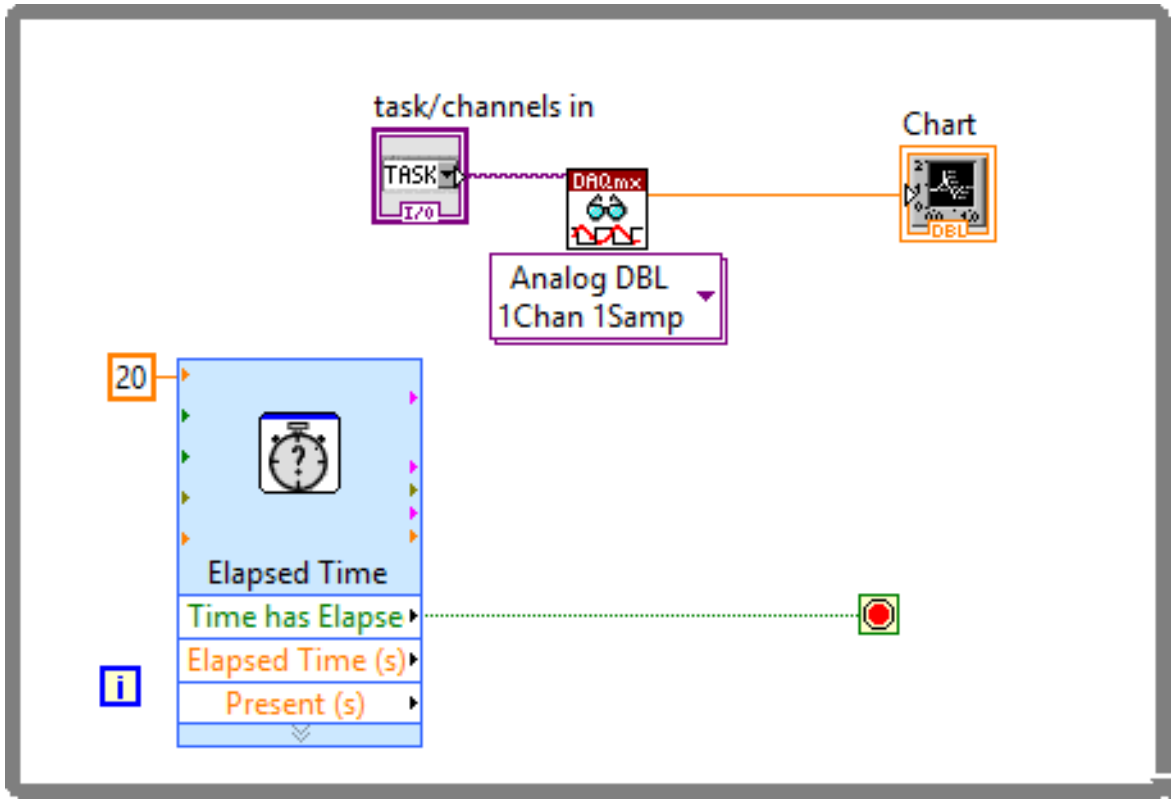




Time Delay – pauzira izršavanje *VI*-a za tačno definisano vreme, slična kao i *Wait* + signal greške



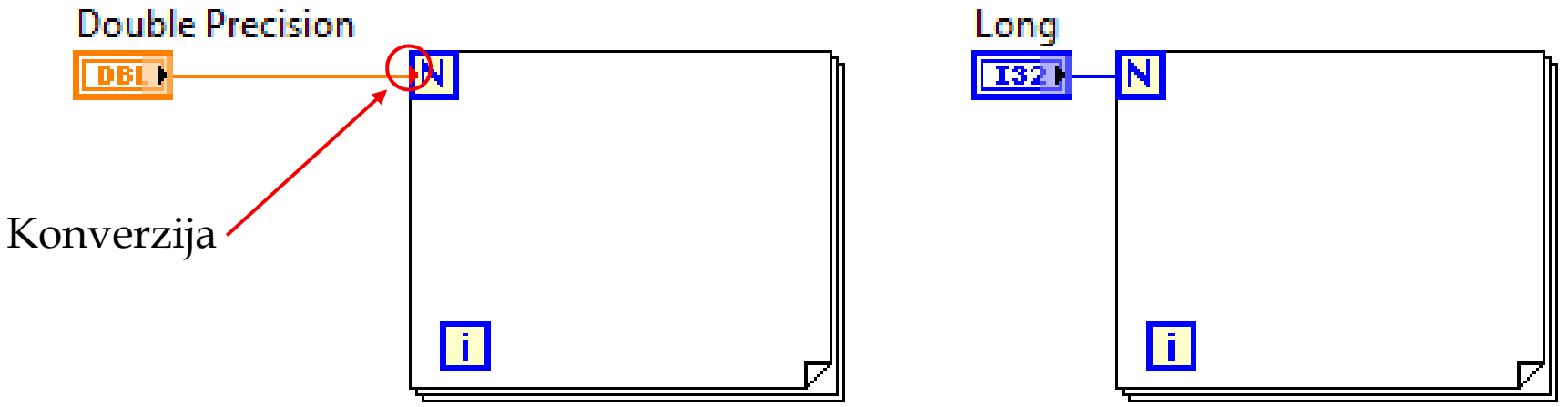
Funkcije za sinhornizaciju



Elapsed Time – za razliku od funkcija *Wait* i *Wait Until Next ms Multiple* ne pauzira izvršavanje petlje do isteka zadatog vremenskog intervala. U svakoj iteraciji proverava da li je proteklo 20 sekundi od prvog poziva funkcije i ukoliko jeste prekida izvršavanje *while* petlje (u gornjem primeru).

Numerička konverzija

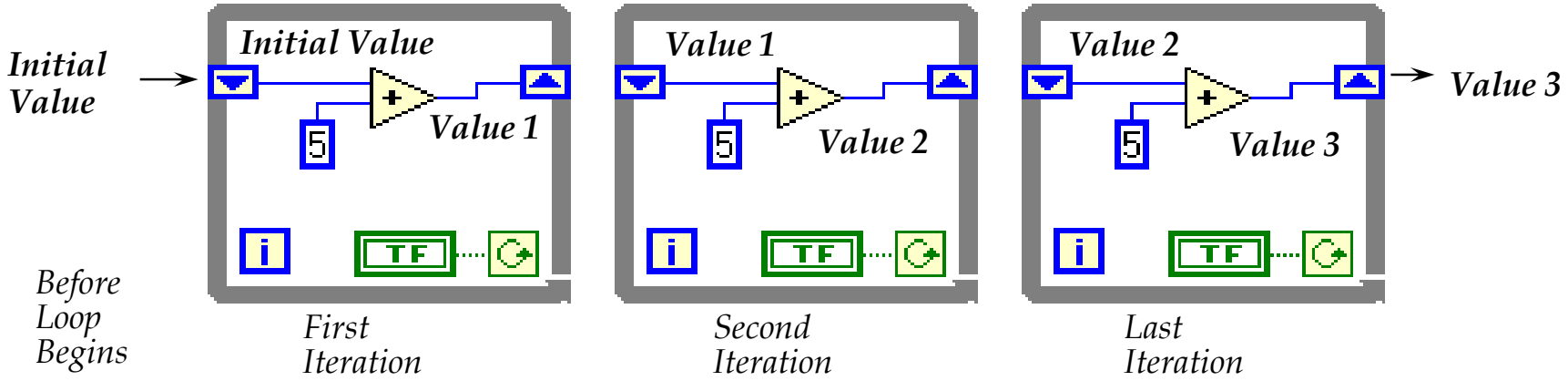
- Kada je potreno LabVIEW automatski vrši numeričku konverziju na reprezentaciju koja zahteva veći broj bita.
- Izuzetak *count terminal* za *for* petlju koji je po definiciji tipa *long*.
- Konverzija se obeležava crvenom tačkom.



- Ukoliko je broj bita isti LabVIEW bira *unsigned* tip podatka.
- Pri konverziji realnih na cele brojeve, LabVIEW zaokružuje na najbliži ceo broj. Ako je broj oblika x.5 zaokužuje se na najbliži paran broj, npr. 2.5 na 2, 3.5 na 4.

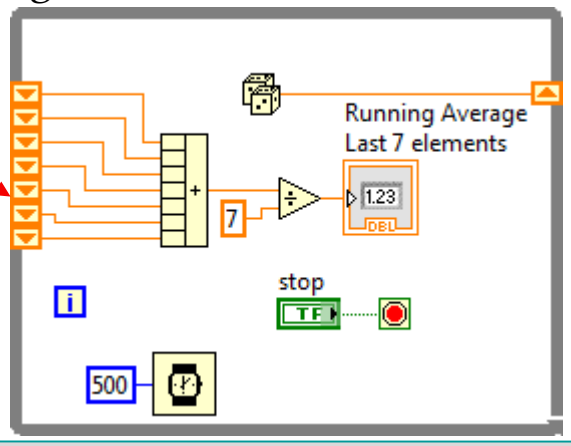
Pristupanje podatku iz prethodne iteracije petlje – *Shift Register*

- *Shift Registar* – desni klik na levu ili desnu ivicu prozora petlje.
- Desni klik i izabrati *Add Shift Register.*
- Desni terminal sadrži vrednost registra na kraju iteracije.
- Levi terminal sadrži podatak na početku iteracije (vrednost registra na kraju prethodne iteracije).



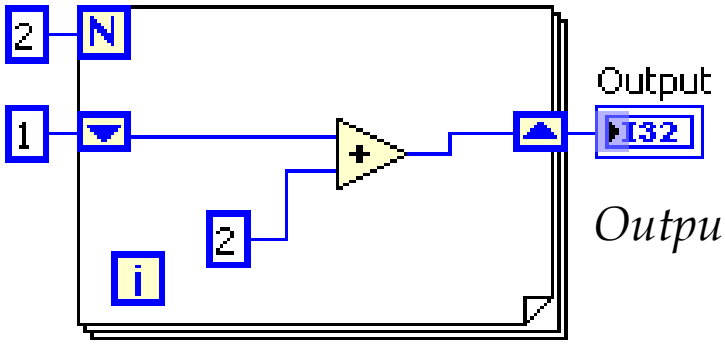
Shift Registar – moguće pristupanje podacima iz nekoliko prethodnih iteracija petlje, npr. realizacije *running average* filtra.

Right-click the left terminal to add new elements

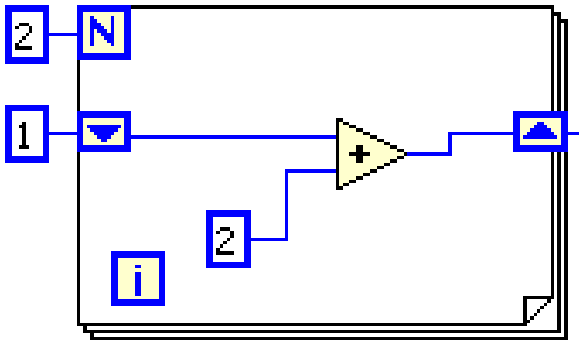


Inicijalizacija *Shift Register*-a

- Inicijalizovani



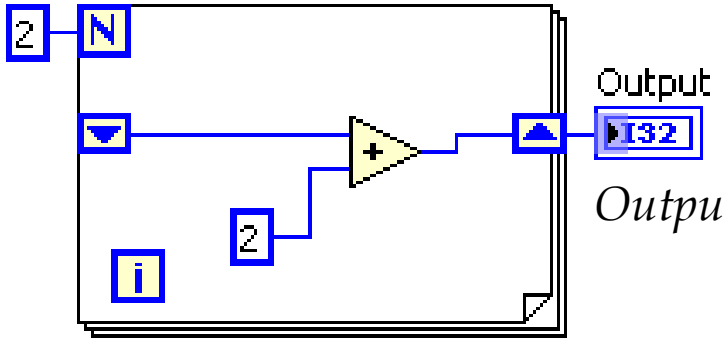
Output = 5



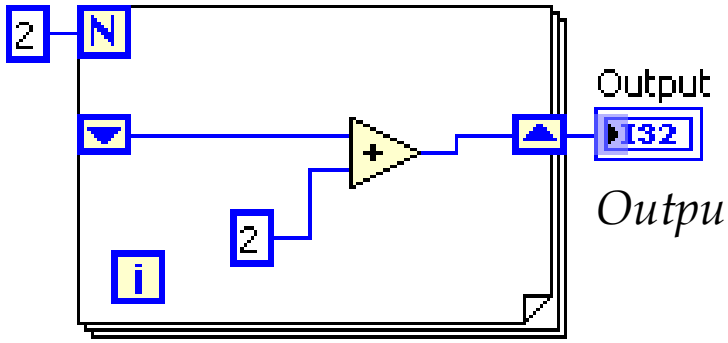
Output = 5

Run Once → *VI stops execution* → *Run Again*

- Nenicijalizovani



Output = 4



Output = 8

Vežba 7

- Proširiti prethodni zadatak tako da se u svakoj iteraciji prikaže srednja vrednost zaokruženih brojeva iz poslednjih pet iteracija.

Nizovi - *Array*

- Kolekcija podataka istog tipa.
- Ne može niz nizova.
- Jedna ili više dimenzija, do 2^{31} elemenata po dimenziji,
- Elementima se pristupa na osnovu indexa, prvi index je 0.

index	0	1	2	3	4	5	6	7	8	9
10-element array	1.2	3.2	8.2	8.0	4.8	5.1	6.0	1.0	2.5	1.7

2D array

	0	1	2	3	4	5	6
0							
1							
2							
3							
4							

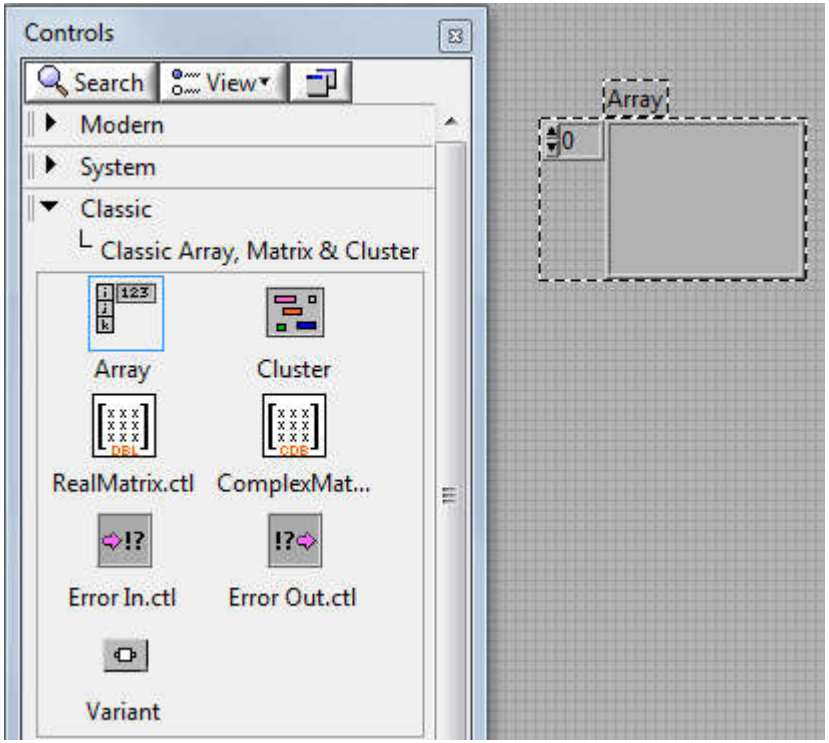
5 redova (*row*) sa po 7 kolona (*column*) = 35 elemenata

Prvo broj reda pa broj kolone.

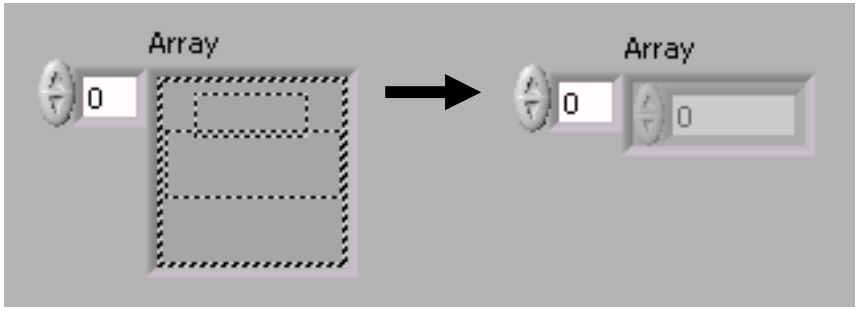
Kreiranje nizova

- ***Front Panel***

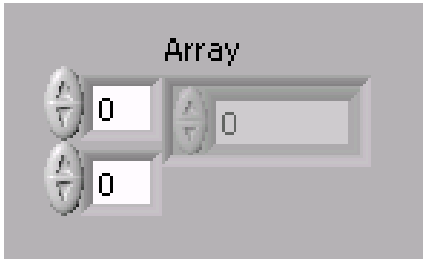
1. Izabrati ***Array*** sa palete *Array, Matrix & Clusters*



2. Postaviti objekat željenog tipa podatka u *Array*. Da li je *Array* kontrola ili indikator zavisi od toga da li je postavljeni objekat kontrola ili indikator (može se promeniti desnim klikom).



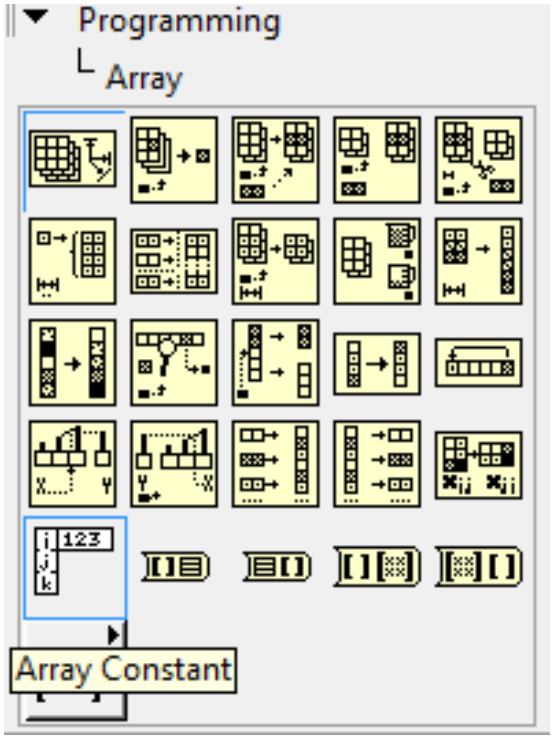
Po potrebi dodati dimeniziju
Add Dimension



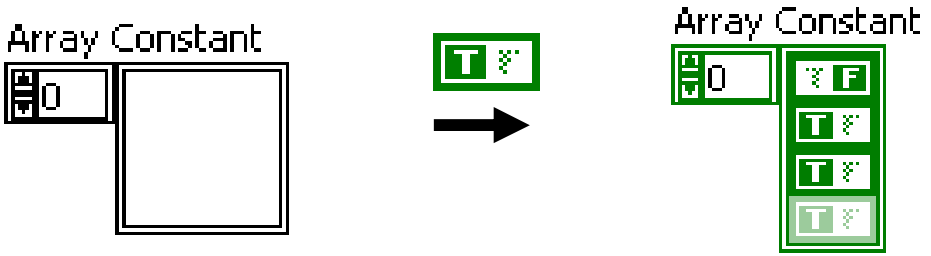
Kreiranje nizova

- ***Block Diagram* – Kreiranje niza konstanti**

1. Izabrati ***Array Constant*** sa palete *Array*.



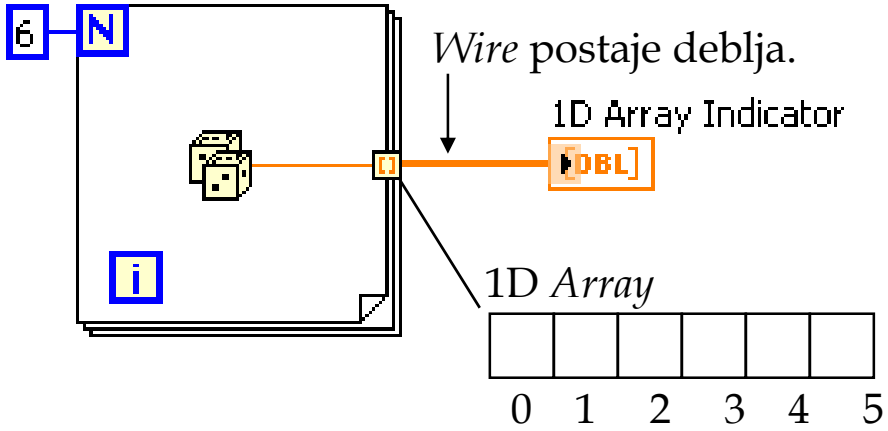
2. Postaviti konstantu željenog tipa podata u *Array*.



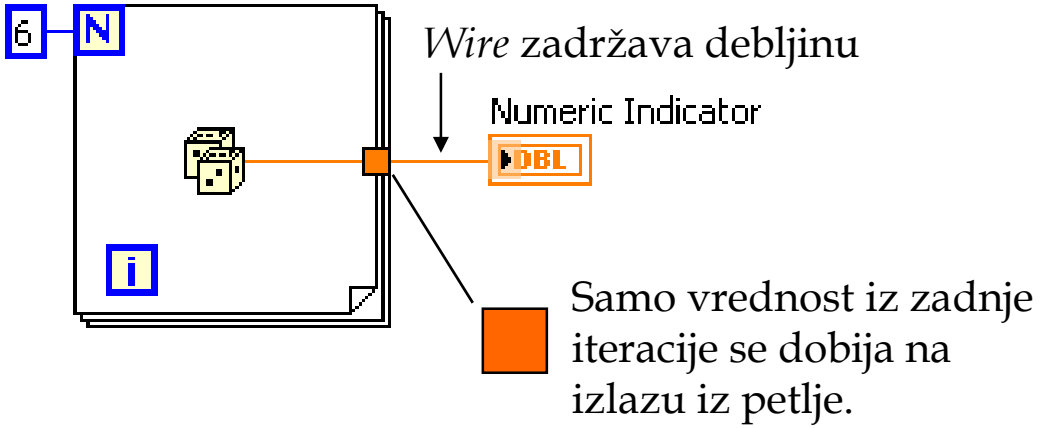
Auto-Indexing

- Petlja može akumulirati niz na granici ukoliko je uključen *auto-indexing*.
- Za *for petlju auto-indexing* je podrazumevan, a za *while* petlju nije, tj. izlaz iz tunela je skalar.
- Desni klik na tunel i *enable/disable auto-indexing*.

Auto-Indexing Enabled

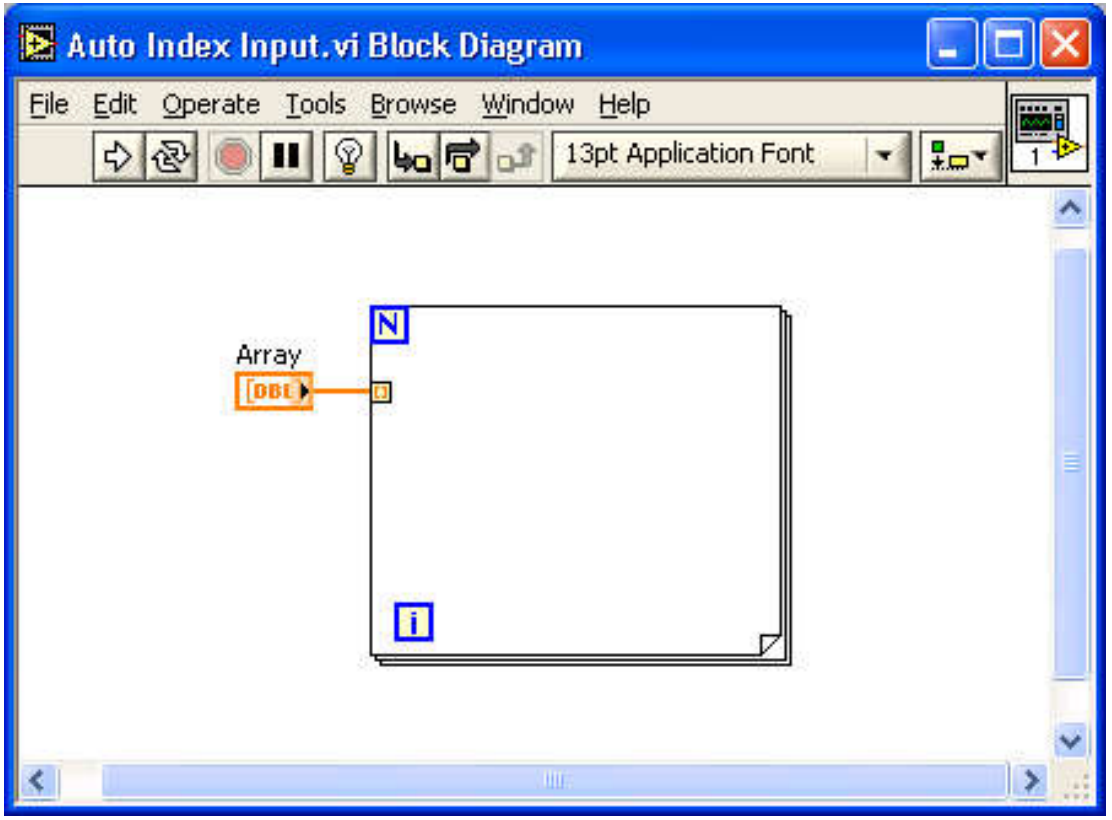


Auto-Indexing Disabled

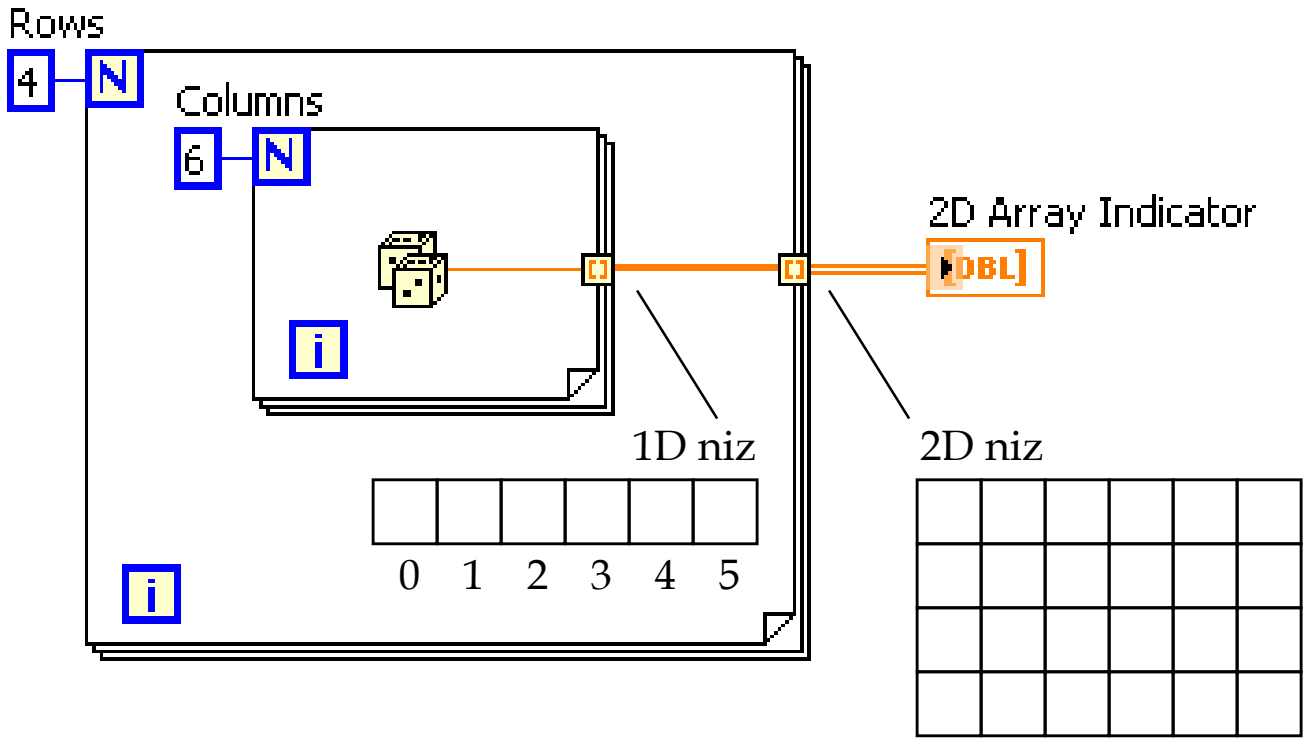


Auto-Indexing

- Niz se može koristiti kao ulaz u *for* petlju i tada je broj iteracija petlje podešen na broj elemenata niza.
- Strelica *Run* nije izlomeljena iako nije povezan *Count Terminal.*

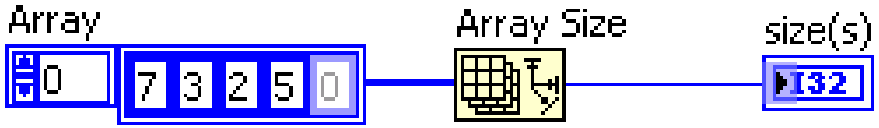


Formiranje 2D niza

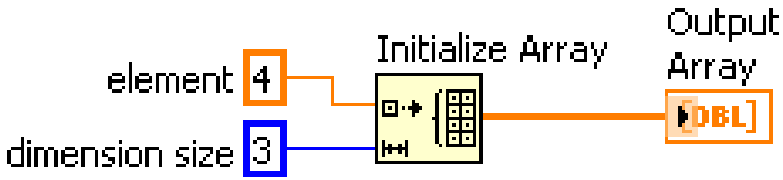


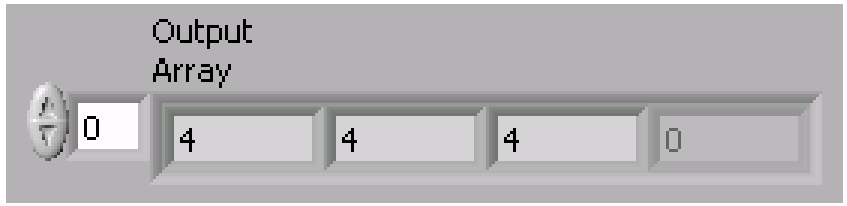
- Unutrašnja petlja formira elemente redove.
- Spoljašnja petlja ih spaja u kolone.

Najčešće funkcije za rad sa nizovima

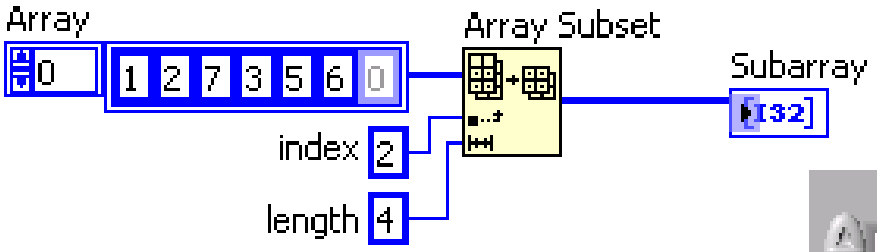


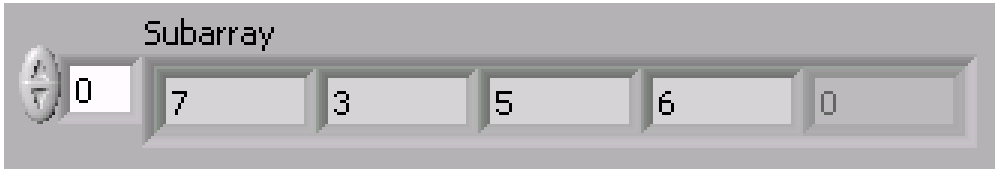
Array Size





Initialize Array

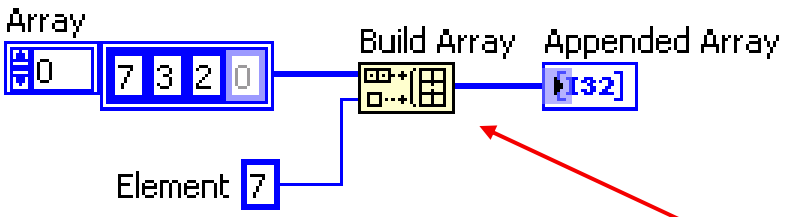




Array Subset

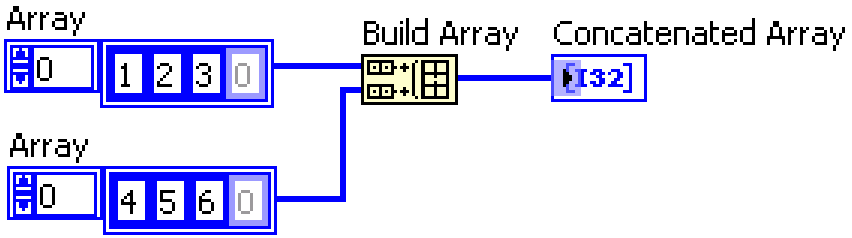
Najčešće funkcije za rad sa nizovima

Build Array funkcija

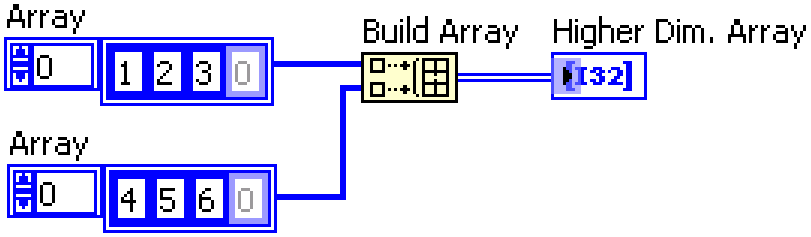


Appending an element

Desni klik i označiti *Concenate Inputs.*



Concatenate Inputs

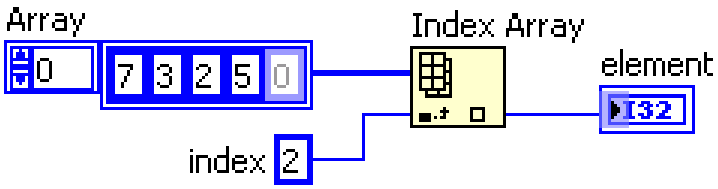


Building a higher dimension array

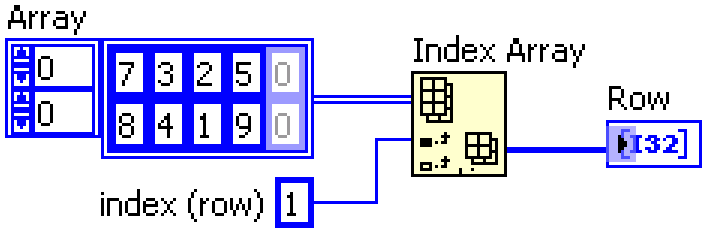
default

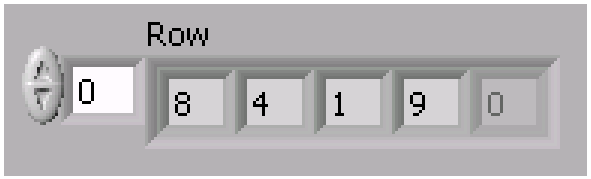
Najčešće funkcije za rad sa nizovima

Index Array funkcija

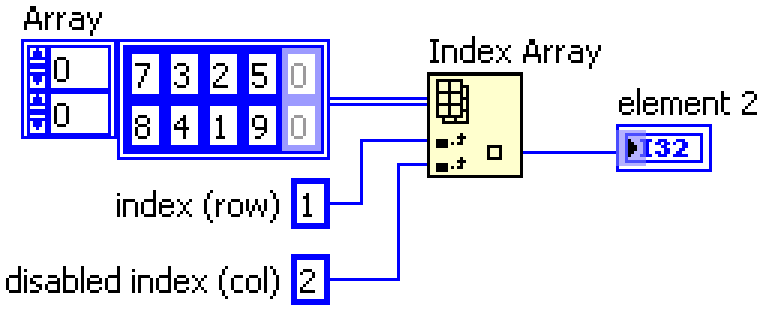


Extracting an Element



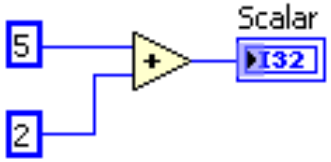
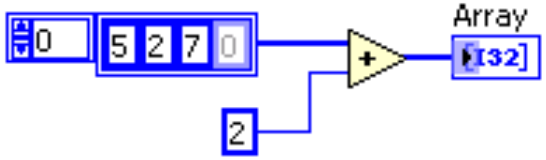
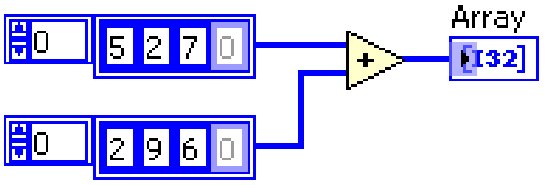
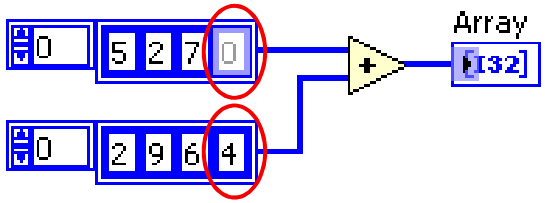


Extracting a Row



Extracting an Element of a Row

Ulazi funkcija mogu biti različitih tipova.
Sve aritmetičke funkcije LabVIEW su polimorfne.

Kombinacija		**Rezultat**	
Skalar + Skalar		Skalar	
Niz + Scalar		Niz	
Niz + Niz		Niz	
Niz + Niz		Niz	

Vežba 8

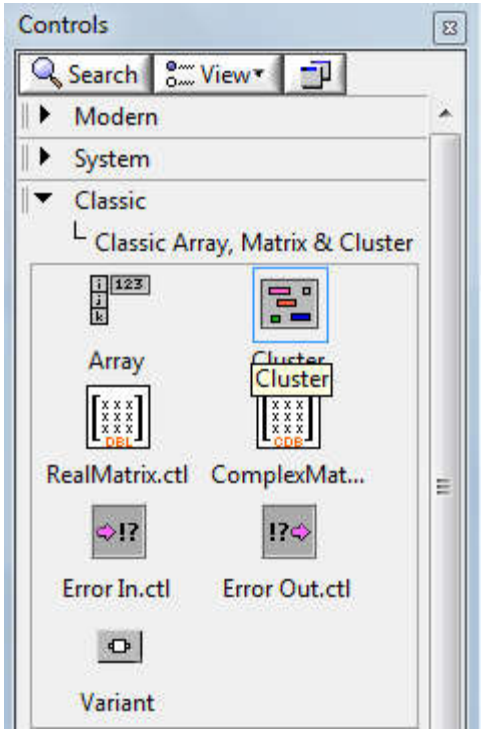
Generisati niz brojeva od 2 do 4 sa korakom od 0.01.

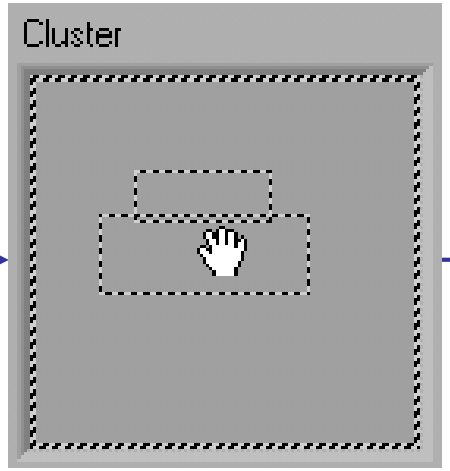
Clusters

- Struktura koja grupiše podatke koji mogu biti različitog tipa.
- Analogni *struct* u *C* ili *record* u *Pascal*.
- Svi elementi moraju biti ili kontrole ili termniali.

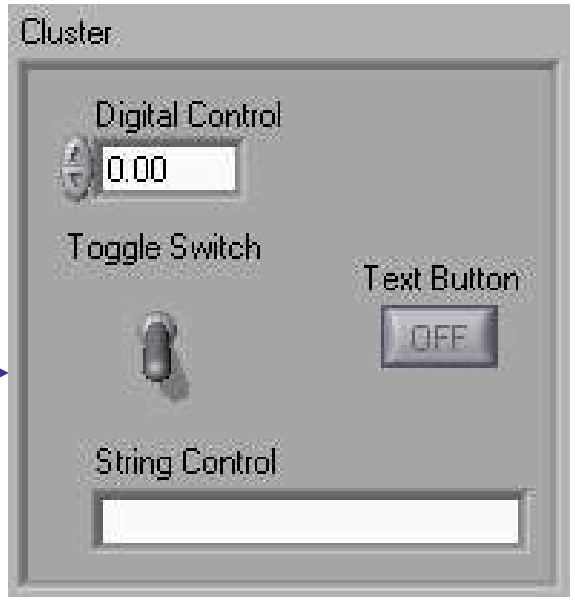
Kreiranje *custer*-a na *Front Panel*-u.

1. Izabrati ***Cluster*** sa palete *Array, Matrix & Clusters*





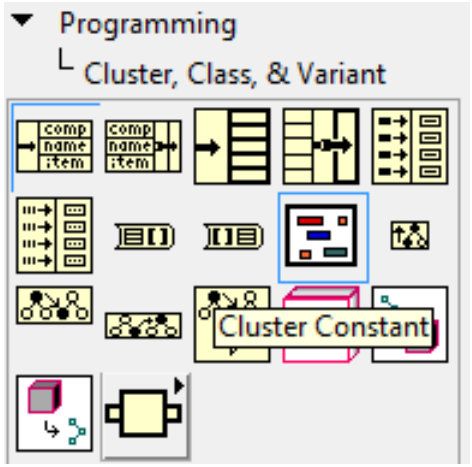
2. Ubaciti objekte unurar *Cluster*-a



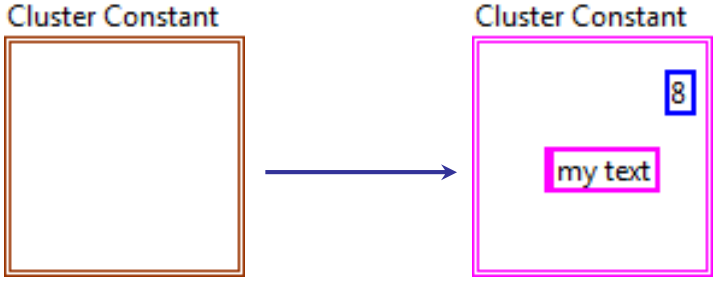
Clusters

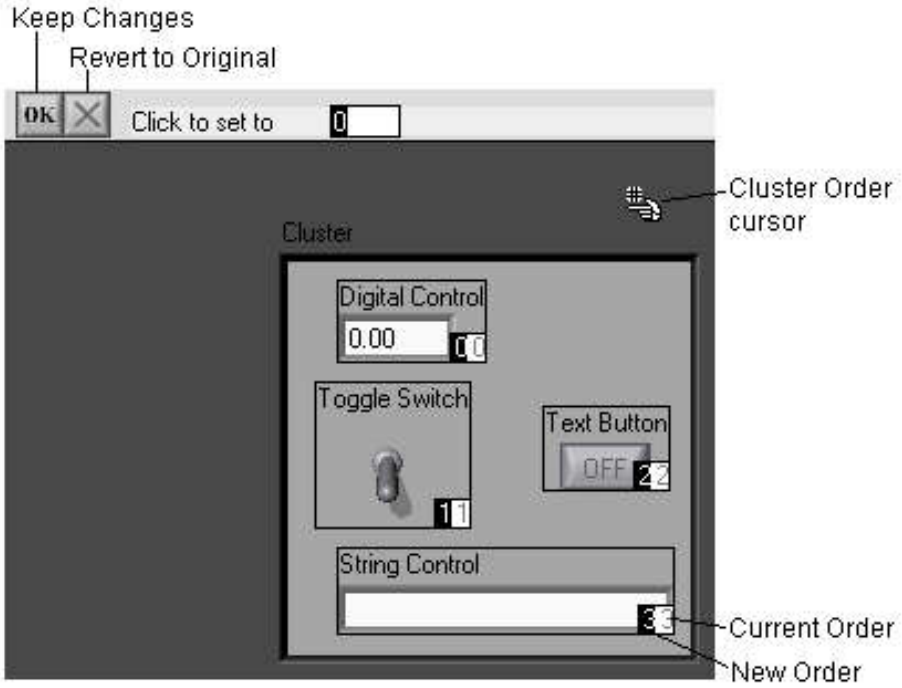
Kreiranje *custer*-a na *Block Diagram*-u.

1. Izabrati ***Cluster Constant*** sa palete *Cluster, Class & Variant*



2. Ubaciti objekte unutar *Cluster Constant*

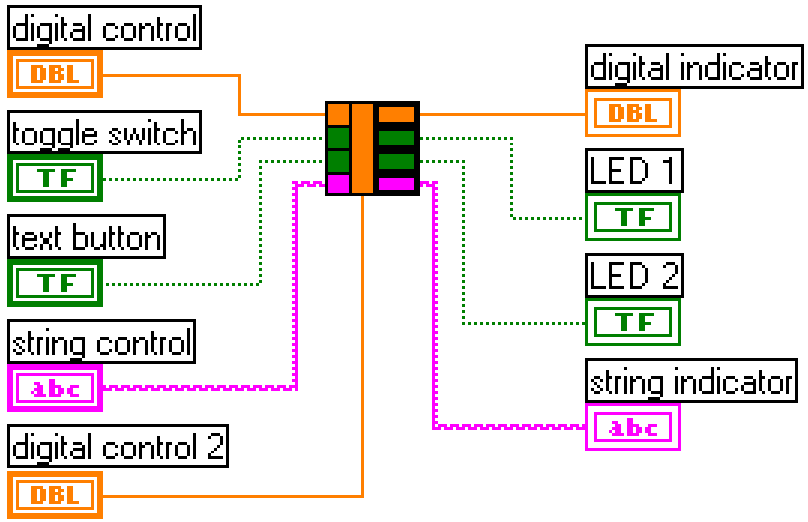


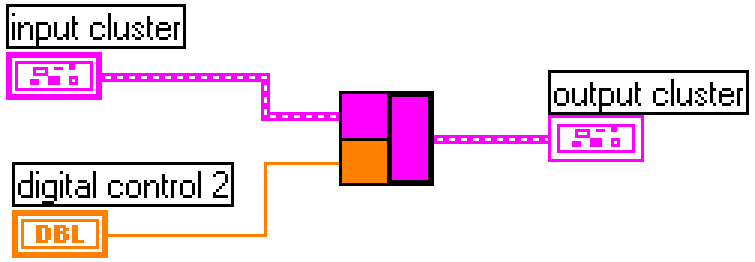


- Elementi unutar *cluster*-a imaju redni broj koji počine sa 0, a koji se može promeniti sa desni klik i izbor ***Reorder Controls in Cluster...***
- Redni broj elementa je bitan kod poređenja dva klastera. LV javlja grešku ukoliko se ne poklapa redosled

Cluster

- *Cluster*-i se mogu upotrebi kada je potrebno preneti nekoliko vrednosti u/iz subVI.
- 28 terminala maximalno za subVI.
- Pojednostavjuje žičenje.





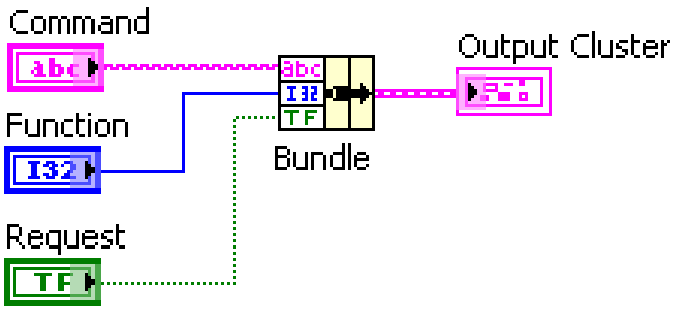
Funkcije za rad sa *Cluster*-ima

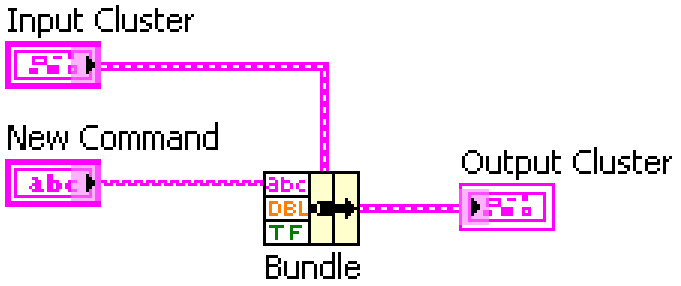
Bundle funkcija

Kreiranje novog *cluster*-a

Izmena postojećeg *cluster*-a

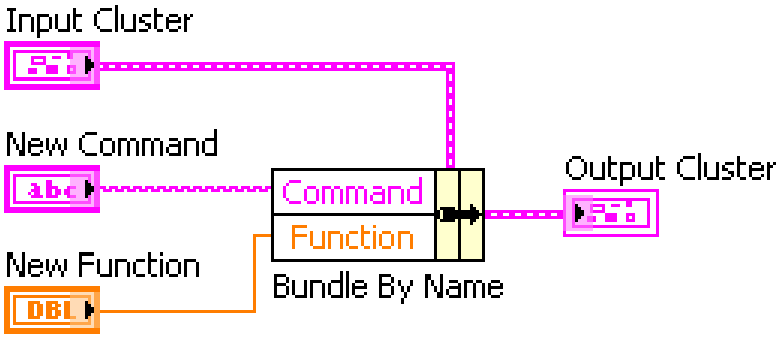
Bundle





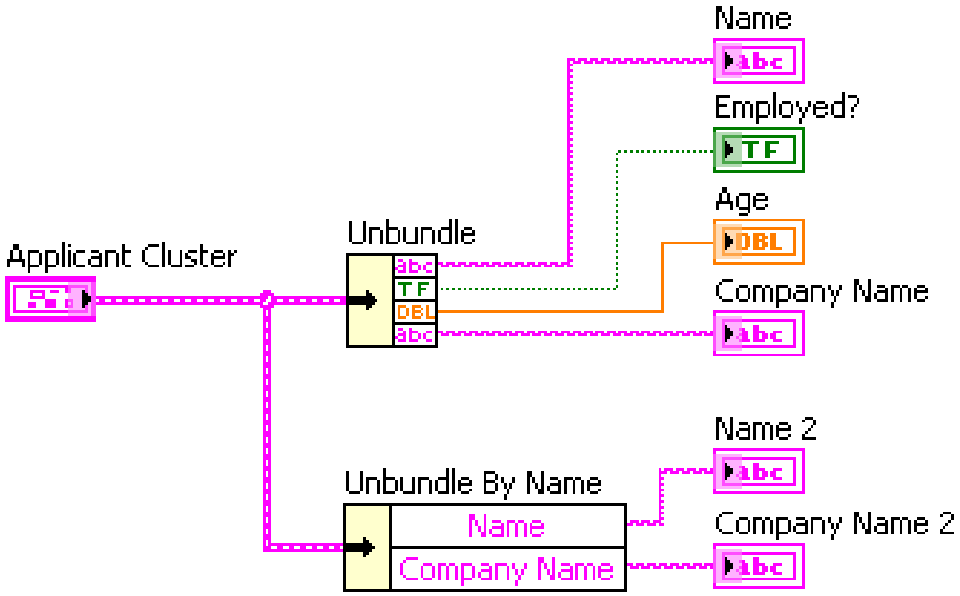
Bundle By Name

Mora postojati definicija postojećeg *cluster*-a ili kao kontrola ili kao indikator da bi ova funkcija mogla da radi, jer zahteva imena polja u *claster*-u.

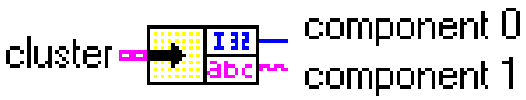


Funkcije za rad sa *Cluster*-ima

Unbundle funkcija



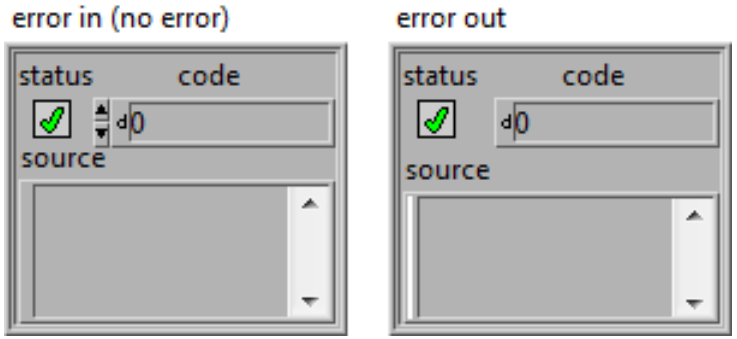
Unbundle

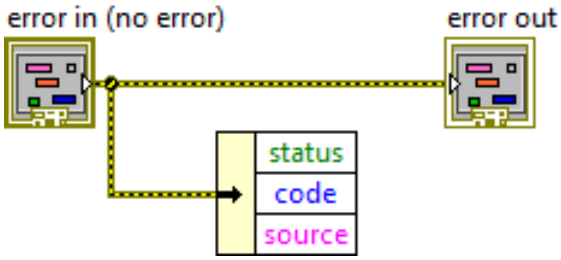


Unbundle By Name

Error Cluster

- *Error Cluster* se može koristiti za *error-handling*.
- Komponente *error clusters* su prikazane na slici.



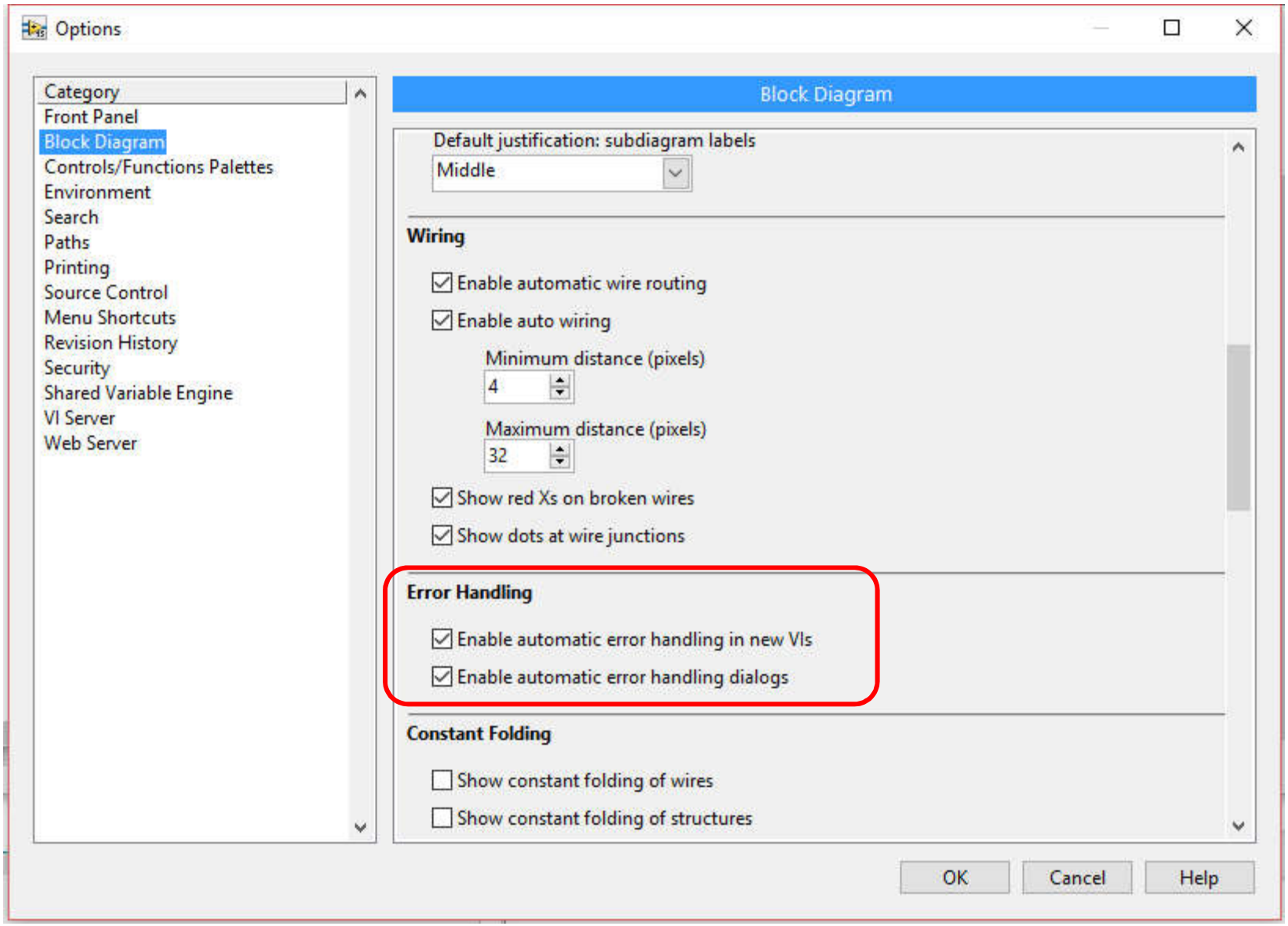


- ***Status*** *is a Boolean value that reports TRUE if an error occurred. Most VIs, functions, and structures that accept Boolean data also recognize this parameter.*
- ***Code*** *is a signed 32-bit integer that identifies the error numerically. A non-zero error code coupled with a status of FALSE signals a warning rather than a error.*
- ***Source*** *is a string that identifies where the error occurred (function, subVIs).*

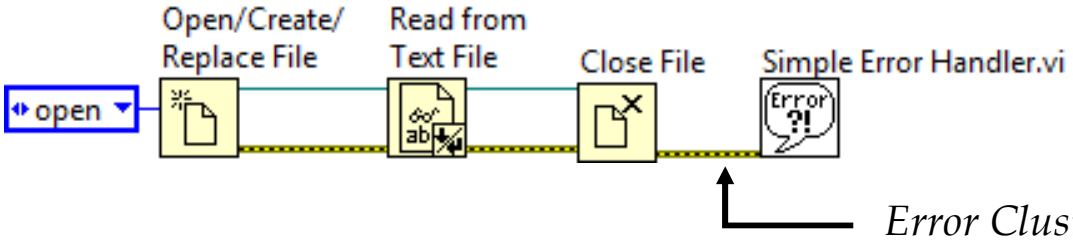
Error-Handling

- *Error-Handling* predstavlja reakciju na pojavu greški koje su uzrokovane pozivanjem nepostojećih resursa (akvizicione kartice ili pristupanje nepostojećem fajlu).
- Postoje dva tipa *Error-Handling*-a:
  - Automatski – U slučaju greške LV (LabVIEW) suspenduje izvršavanje (ostavlja se korisniku da odluči da li da nastavi ili obustavi izvršavanje), prikazuje čvor (*blinking*) u kome je došlo do greške i izbacuje obaveštenje o grešci.
  - Manuelni - programski se obezbeđuje reakcija na grešku. Moguće je nastaviti kod sa obaveštenjem o grešci, preuzeti aktivnosti kako bi se otklonila greška (npr. da korisnik proveri da li je akviziciona kartica povezana) ili da se potpuno igrnoriše greška (nikako se ne preporučuje, jer može izazvati blokadu izvršavanja programa, pa čak i "rušenje" operativnog sistema).
- Ukoliko nema manuelnog *Error-Handling*-a, može se isključiti i automatski i tada se korisnik ne obaveštava o pojavi greške, već se greška ignoriše. Naravno, ovo se ne preporučuje jer ne postoji kontrola pojave greški, a time se ne zna ni kakav će biti rezultat daljeg izvršavanje programa.
- Automatski *Error-Handling* se isključuje na: *Tools* » *Options*, zatim se bira kategorija *Block Diagram*, i u *Error-Handling* de selektuje se ponuđene opcije.

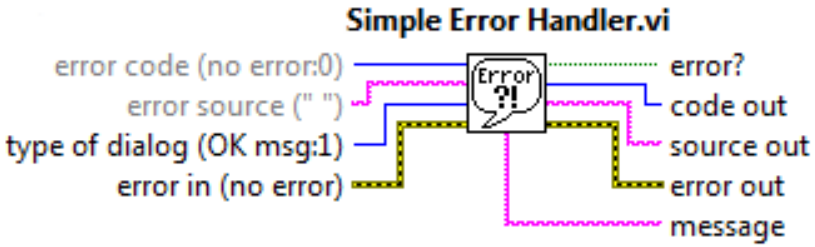
Error-Handling

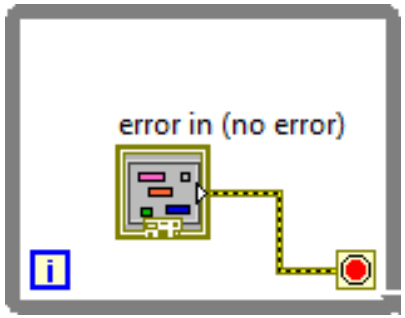


Error-Handling with Error Clusters



- Greška prati *dataflow* model, tj. ako subVI poseduje *error in* i *error out* terminale, *error cluster* prenosi informaciju o pojavi greške.
- Preporuka je da se signal greške (*error cluster*) povezuje kod svih subVI koje poseduju *error in* i *error out* terminale.
- Za *Error-Handling* može se koristi *Simple Error Handler* koji se nalazi na paleti *Dialog & User Interface*.
- *Simple Error Handler* obaveštava korisnika o grešci, kod greške i njenom izvoru. Prethodno važi ukoliko se koristi signal greške iz VI-a koji su deo LV-a ili ukoliko je *Error Cluster* programski obezbeđen za ostale VI-eve.

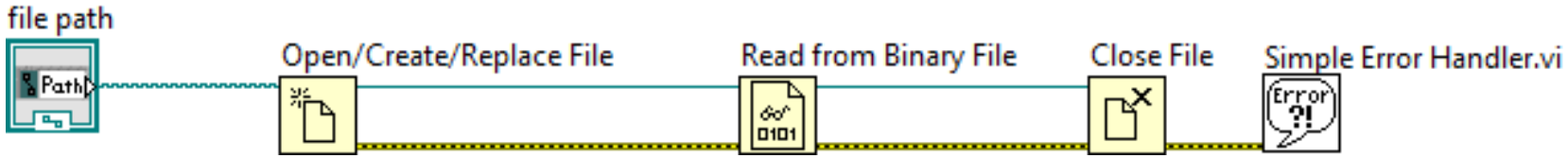




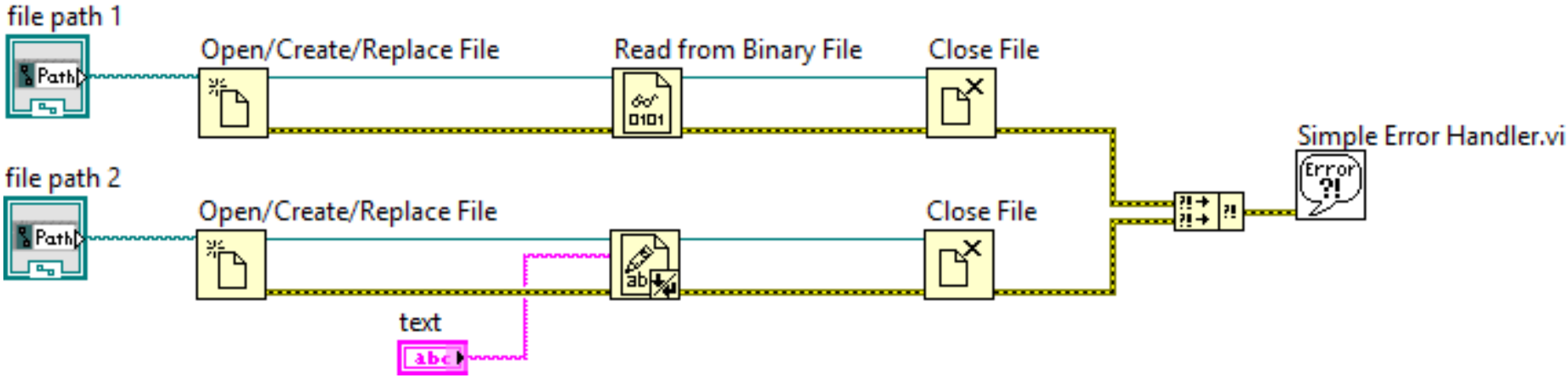
Prekid izvršavanja *while* petlje kada se javi greška (*status* = *true*)

Error-Handling with Clusters

- Ukoliko postoji samo jedan tok signala greške (u smislu *data-flow* programskog modela), signal greške (*Error Cluster*) sadrži podatke o prvoj grešci koja se pojavila, jer se ostali VI-evi koji se nalaze duž toka signala greške neće ni izvršiti.



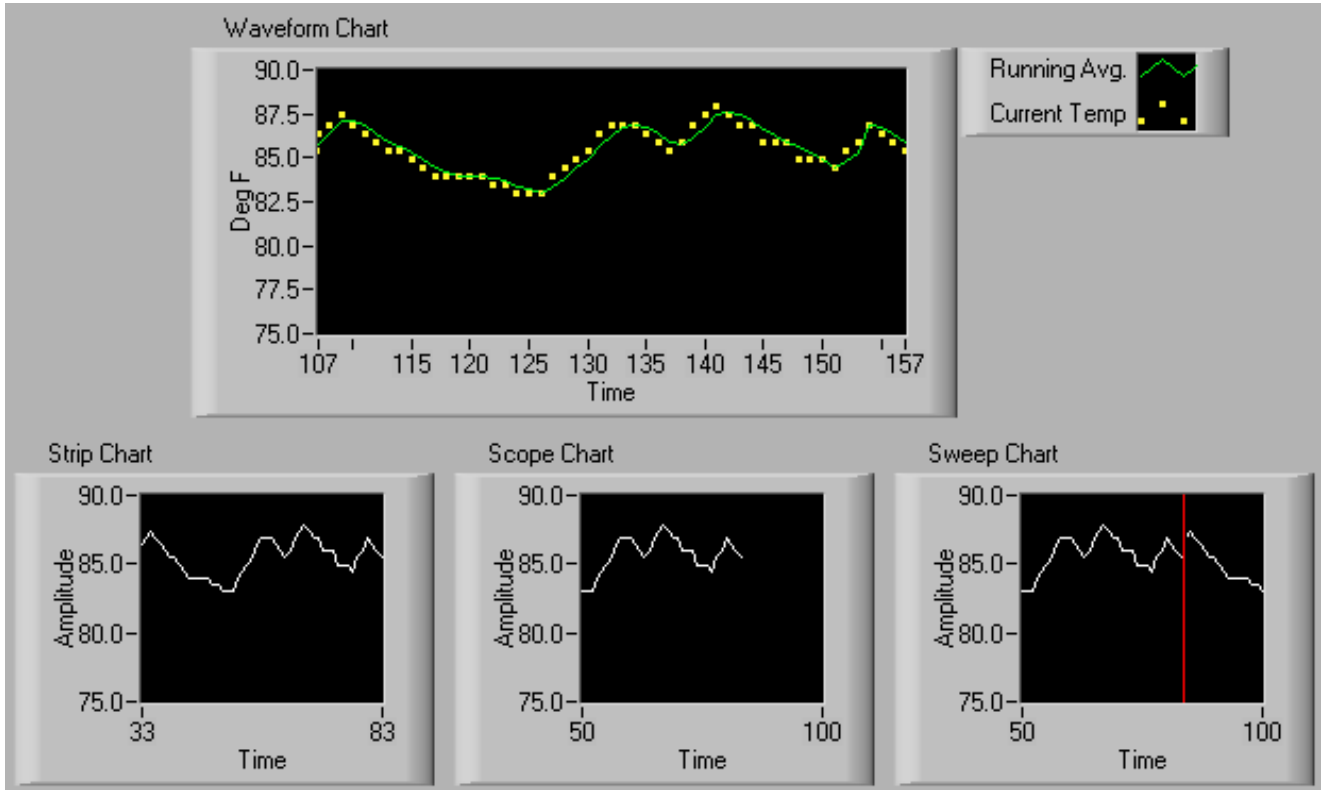
- Za slučaj da postoji više toka signala greške, oni se mogu spojiti korišćenjem funkcije *Merge Errors*.



- Fukcija *Merge Errors* vraća prvu grešku koju pronađe, posmatrano od prvog ulaza na vrhu same funkcije. U slučaju da postoji više greški ne vrši se spajanje stringova koji opisuju svaki od greški u zajednički string. Ako se ne pronađe greška vraćaju se i upozorenja (Warnings), ukoliko ih ima.

Grafičko prikazivanje numeričkih podataka – *Waveform Chart*

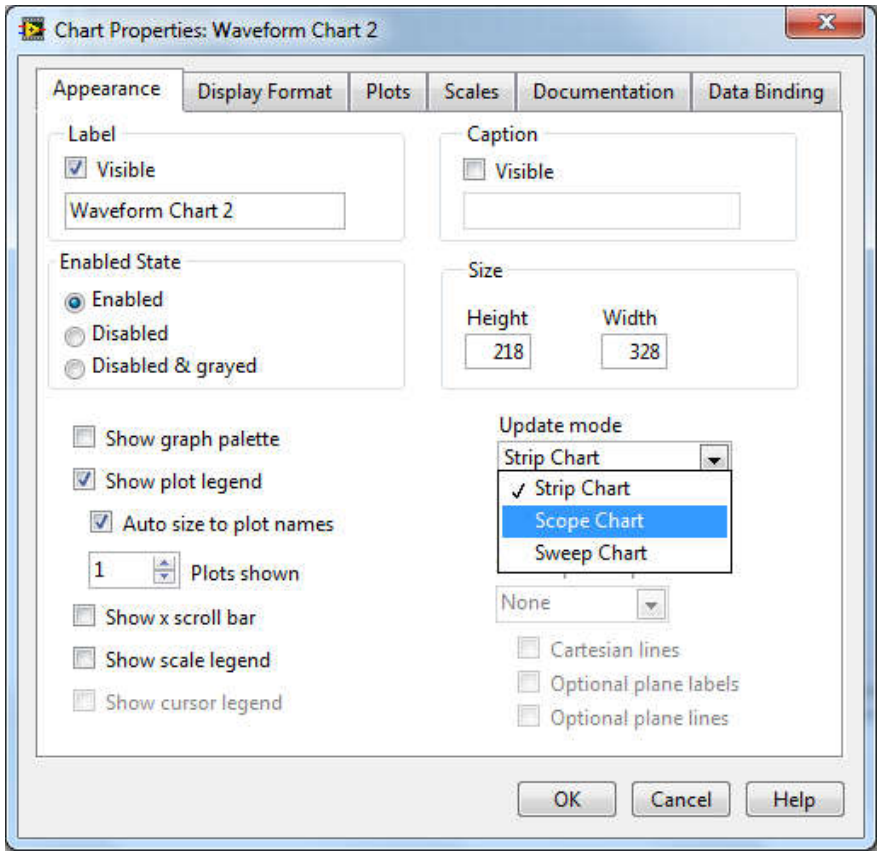
Na paleti *Modern* » *Graphs*



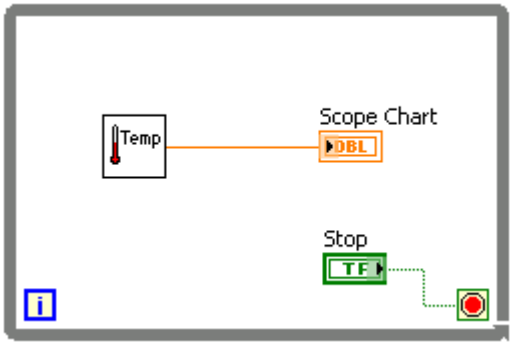
- *Strip Chart*: Kada iscrtavanje stigne do desne ivice, nove vrednosti se dodaju sa desne stane pri čemu izgleda kao da se grafik kreće ulevo.
- *Scope Chart*: Kada iscrtavanje stigne do desne ivice, grafik se briše i iscrtavanje ponovo počine sa leve strane.
- *Sweep Chart*: Kao *Scope Chart*, ali se grafik ne briše, već marker (verikalna linija) prikazuje dokle se stiglo sa icrtavanje novi vrednosti. Levo od markera su nove, desno stare vrednosti.

Waveform Chart

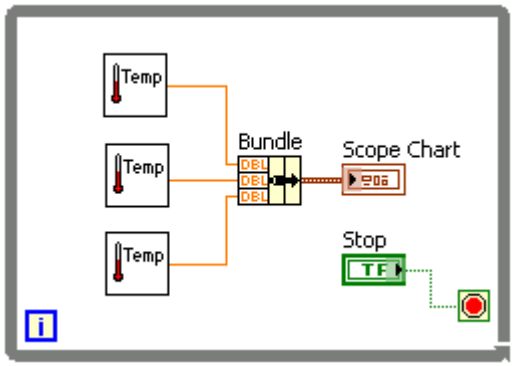
- Izbor moda rada *Chart*-a: desni klik i *Properties*.



Single-Plot Chart



Multiple-Plot Chart

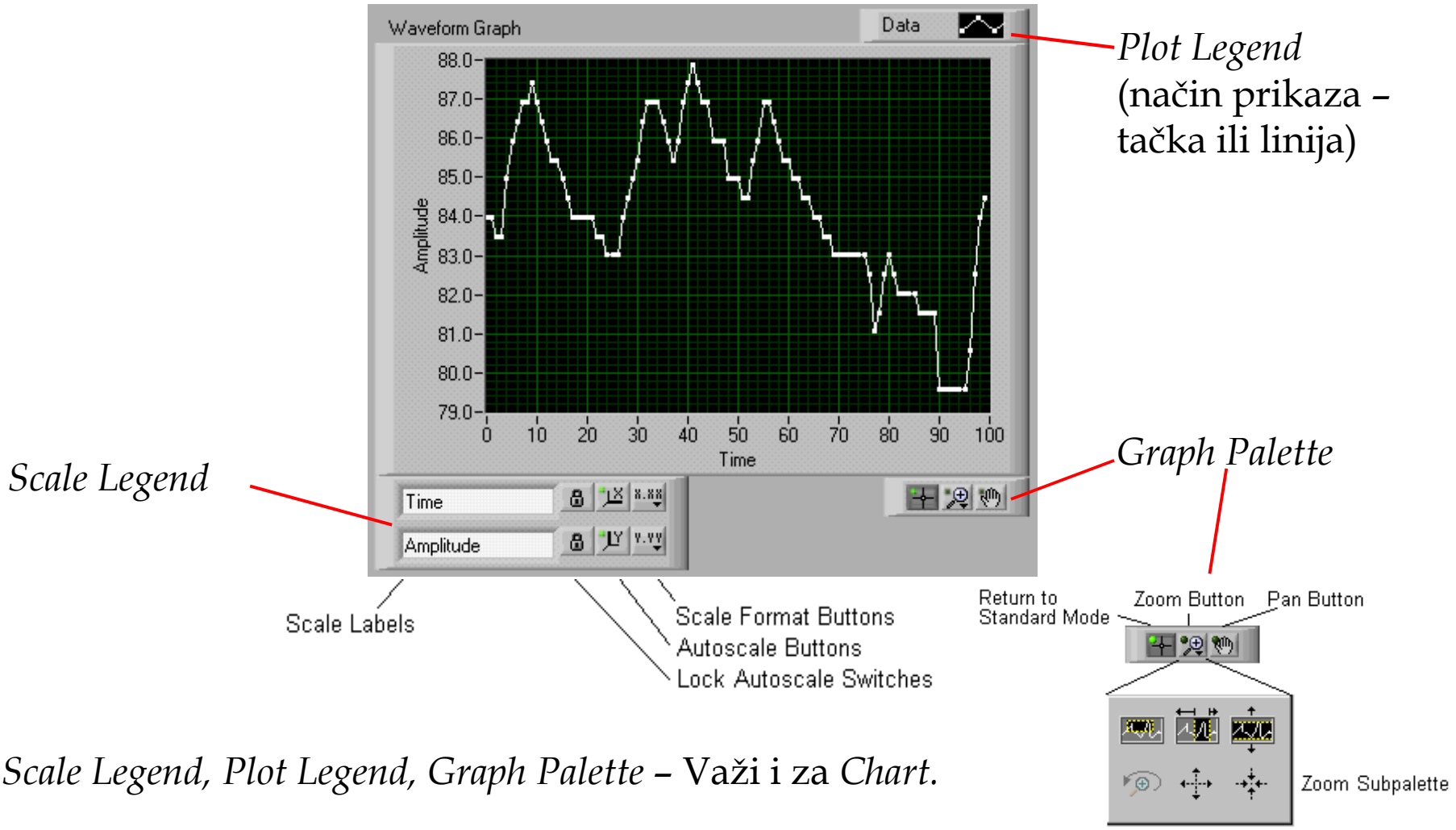


Dodatne opcije:

- Promena izgleda.
- Promena formata i broja decimalnih cifara.
- Promena načina iscrtavanja grafika.
- Promena izgleda i naziva osa.
- Dokumentacija *chart*-a.

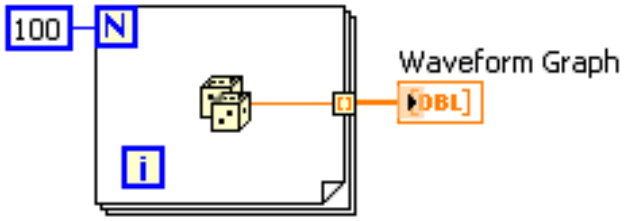
Waveform Graph

- *Waveform Graph* – Prikaz podataka niza u odnosu na indeks u nizu.
- *XY Graph* – Prikazuje jedan niz u odnosu na drugi.

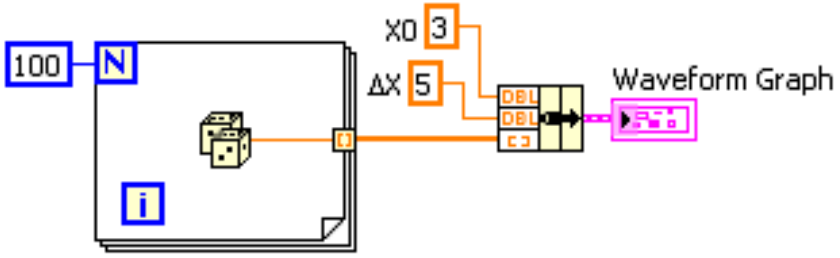


Scale Legend, Plot Legend, Graph Palette – Važi i za *Chart*.

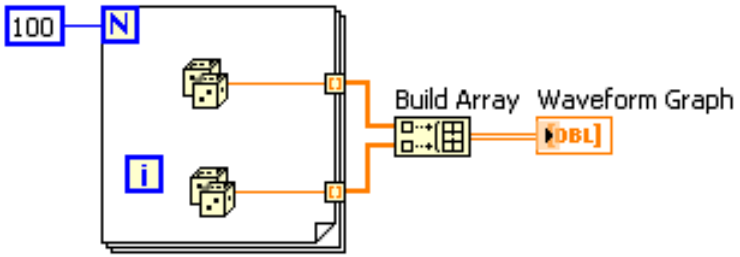
Waveform Graph



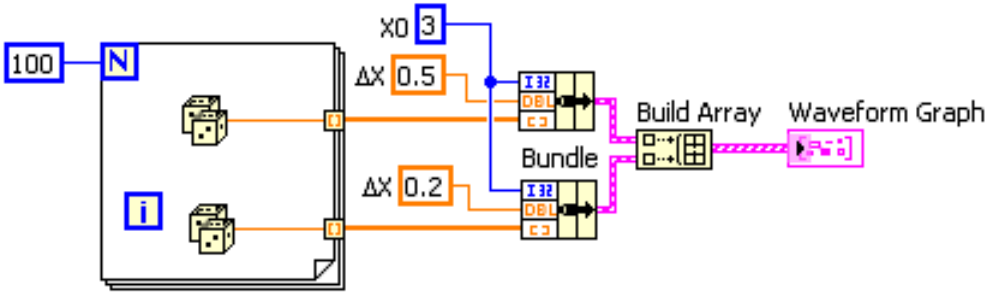
Uniform X axis
Initial X = 0.0
Delta X = 1.0



Uniforma X osa
korisnik specificira
početnu tačku i rastojanje



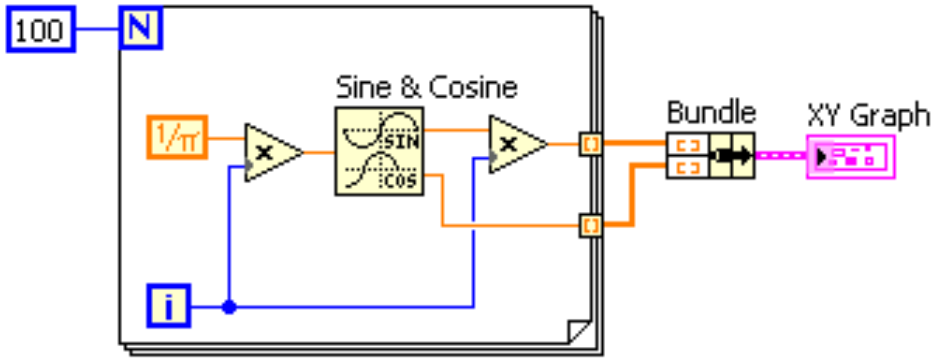
Svaki red u 2D nizu brojeva je
nezavisan diagram:
Početno X = 0
Delta X = 1

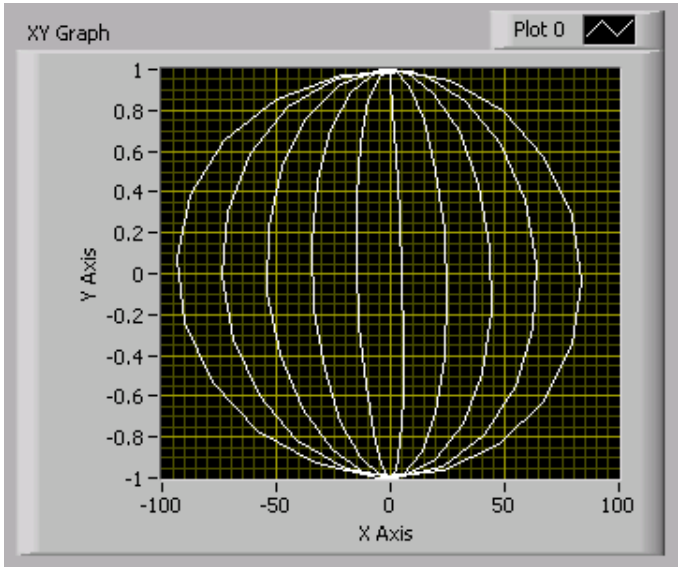


Niz cluster-a, svaki cluster sardži
niz brojeva koji se prikazuju na
Graph-a kao početno X i Delta X.

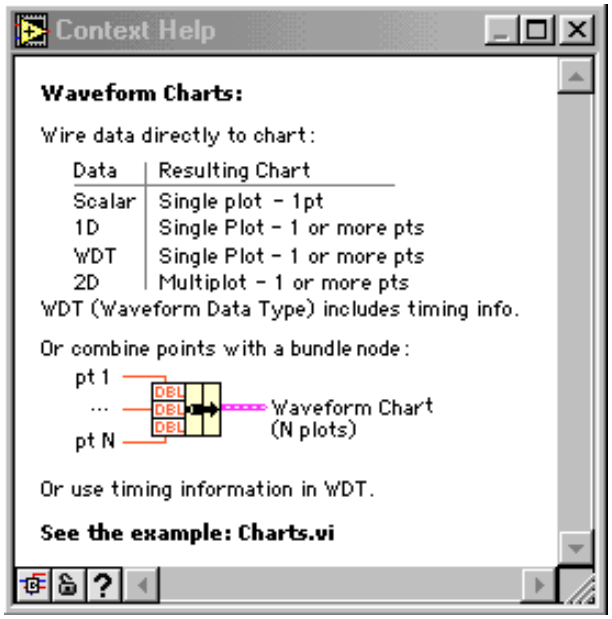
XY Graph

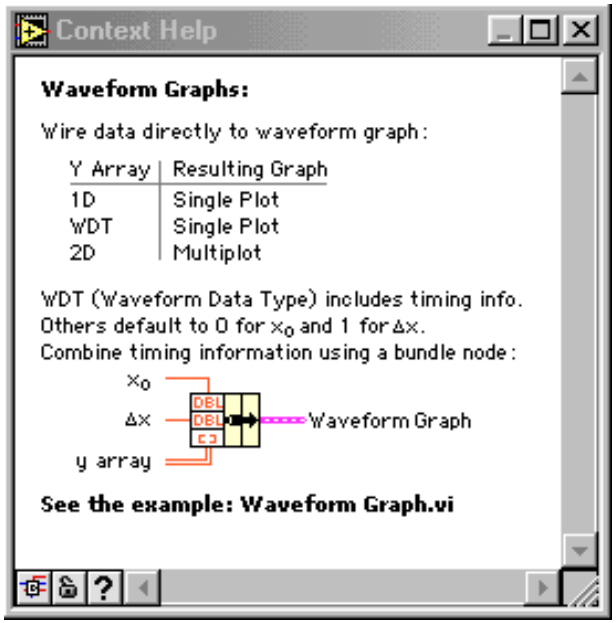
- Neuniformna X ose.
- X i Y koordinate date su nezavisnim nizovima

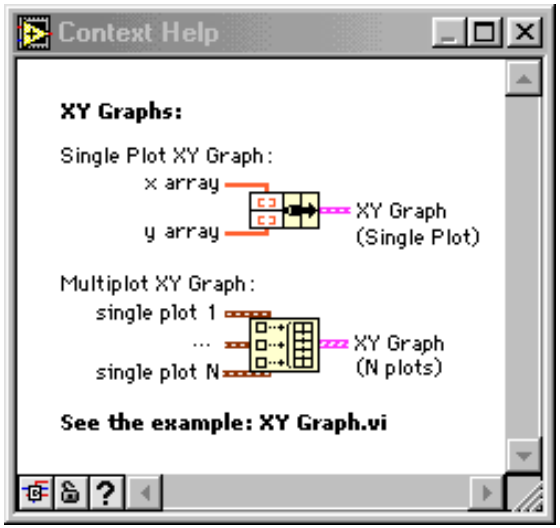




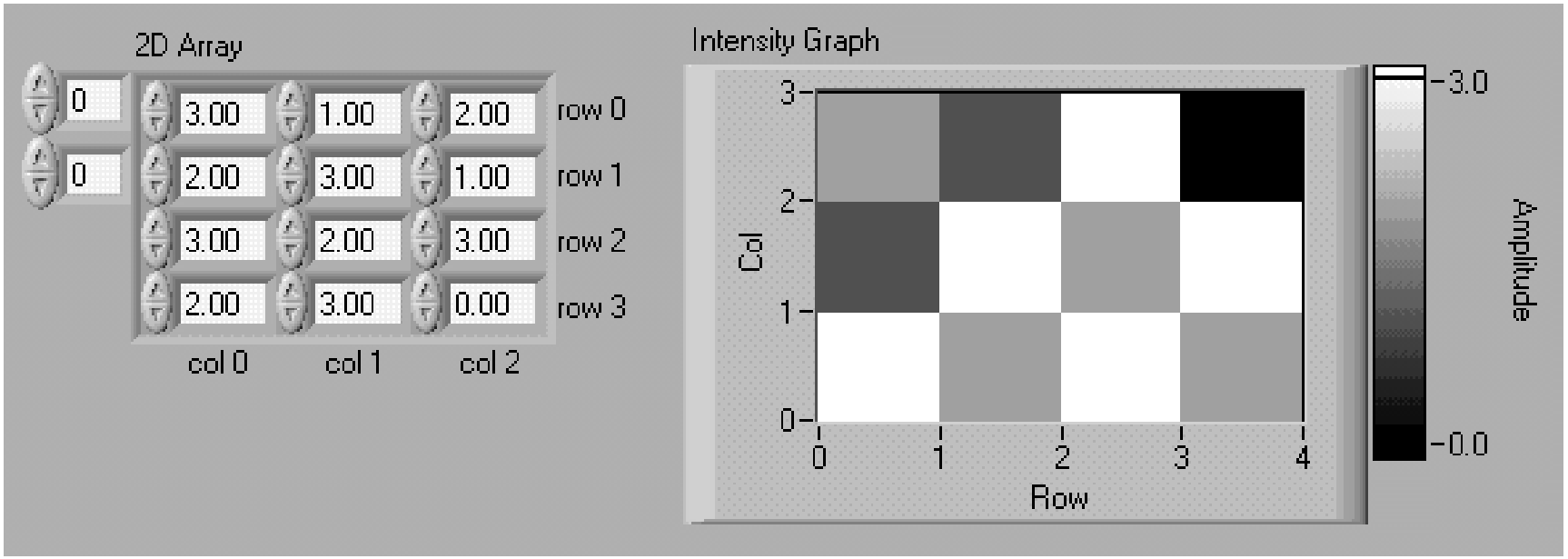
- Context Help olakšava razlikovanje grafika.







Intensity Graph and Chart



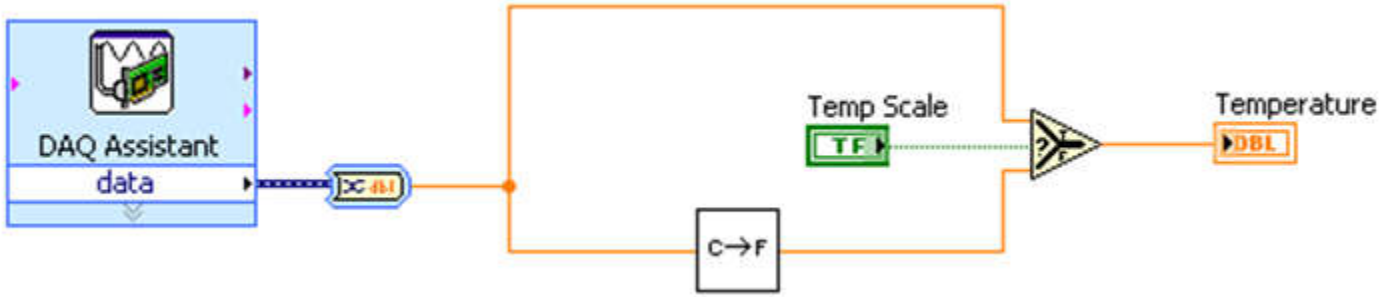
- Korisni za prikazivanje terena, temperaturne raspodele, spektralne analize, procesiranje slike.
- Ulaz je 2D niz, gde element niza predstavlja boju.

Vežba 9

- Nastavak vežbe 8: dodati *Waveform Chart* i *Wavefrom Graph*.

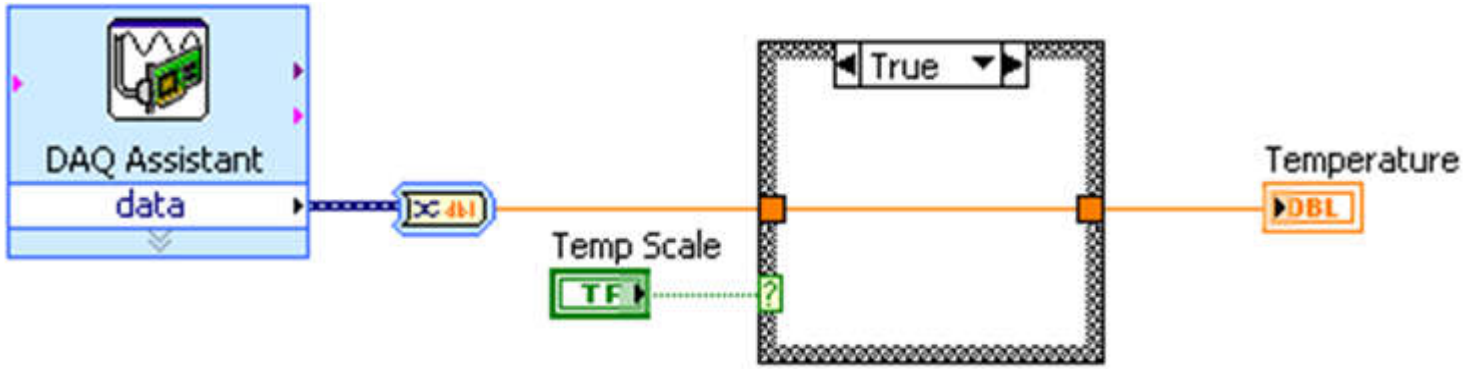
Odlučivanje: *Select Function* i *Case Structures*

- *Programming » Comparison Palette » Select*
- *If Temp Scale is TRUE, pass top input;*
  if temp scale is FALSE, pass bottom input.



- Ukoliko je potrebno izvršiti neliko naredbi za *True* ili *False* slučaj, koristi se *Case Structure*, (*Programming » Structures Palette*).

- *If Temp Scale is TRUE, execute True case;*
  if temp scale is FALSE, execute False case.

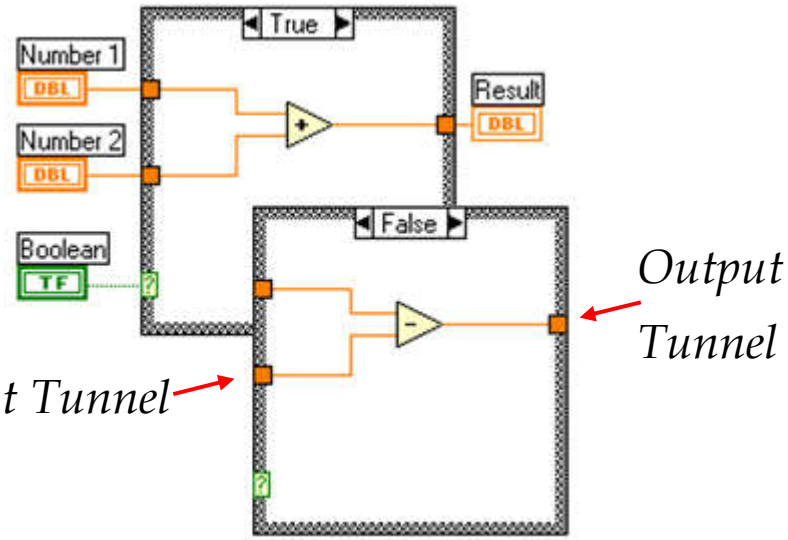


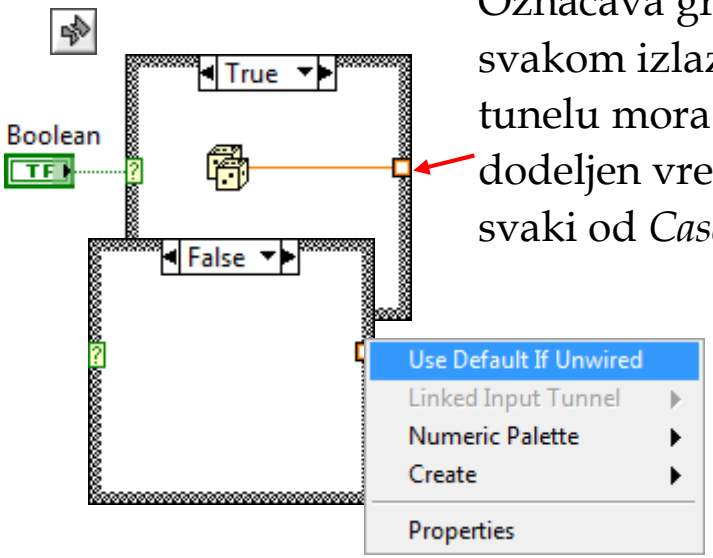
Vežba 10

- Napraviti program koji za zadati broj određuje kvadratni koren ako je veći od nule, a ako je manji vraća njegov kubni koren.

Case Structures

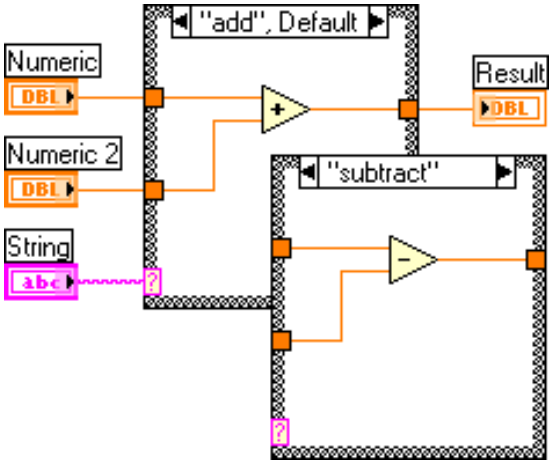
Boolean Case



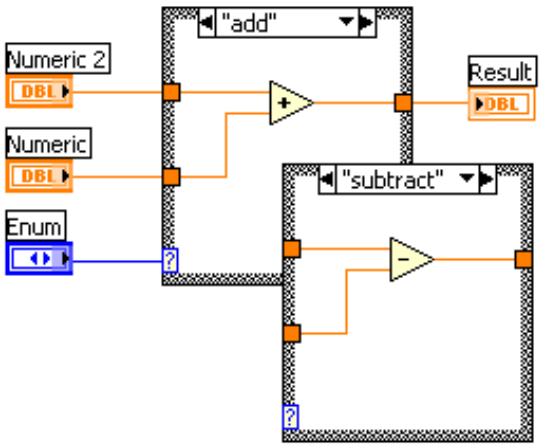


Označava grešku, jer svakom izlaznom tunelu mora biti dodeljen vrednost za svaki od *Case*-ova.

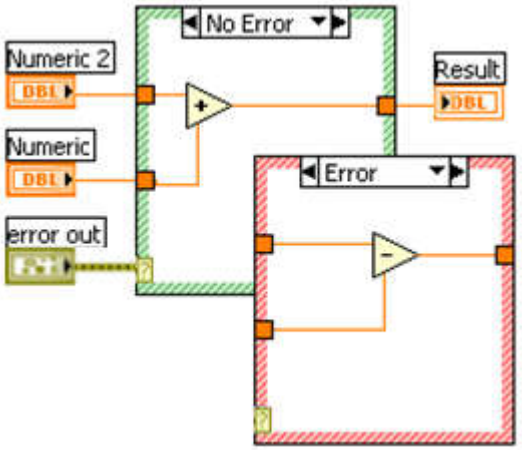
String Case



Enum Case

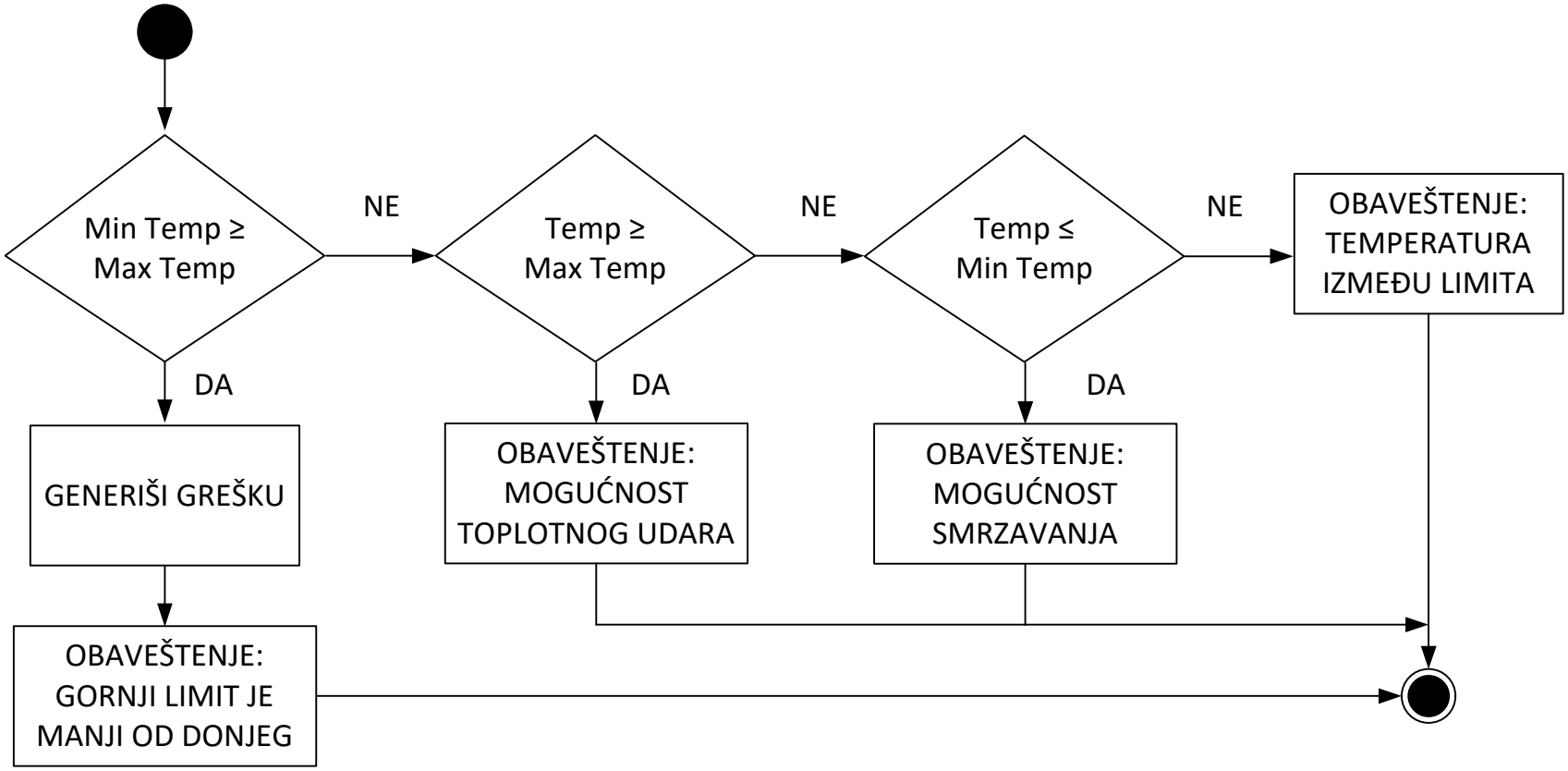


Error Case



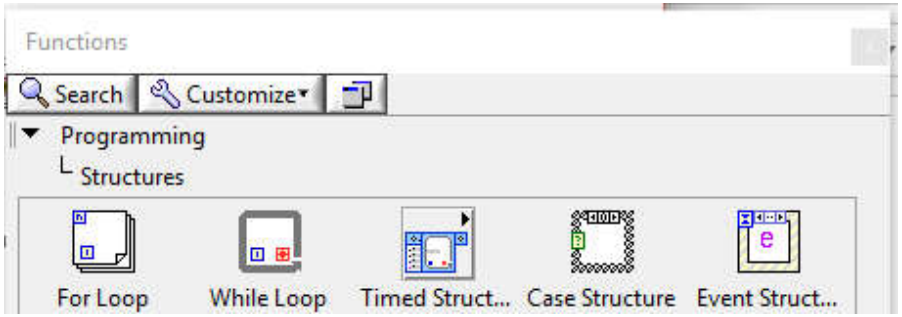
Vežba 11

- Realizovati program koji se izvršava prema sledećem dijagramu toka:



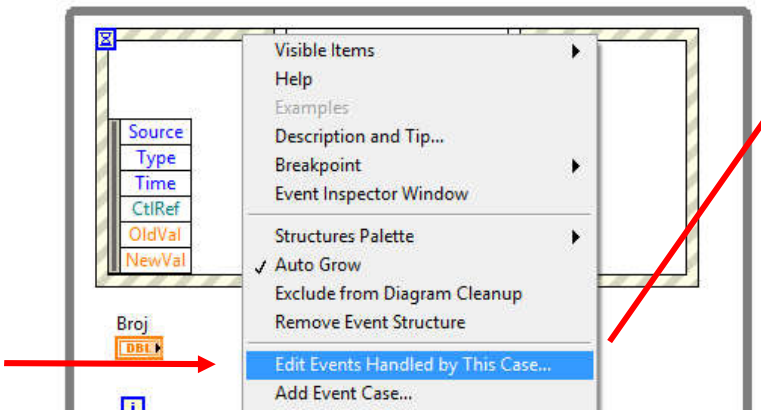
Event Structure

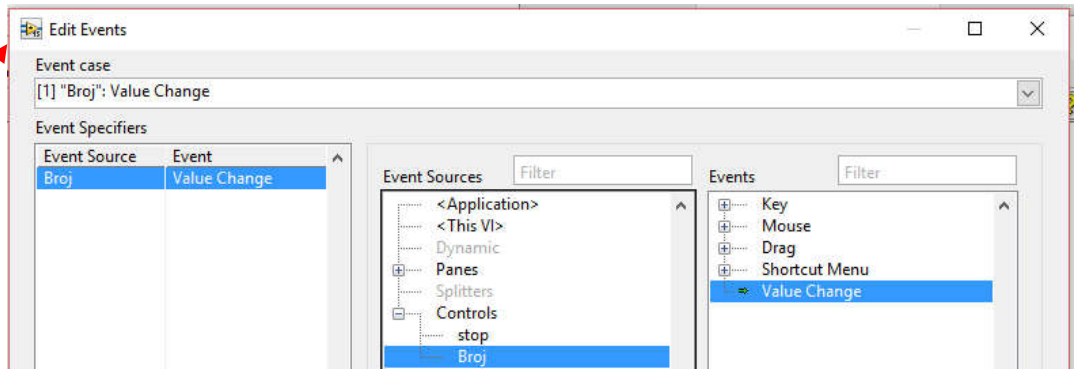
- *Event-driven programming* – metod programiranja koji omogućava komunikaciju između korisnika i korisničkog interfejsa (UI – *User Interface*). Prate se aktivnosti korisnika (aktiviranje određenih tastera, izbor sa padajućeg menija, promena vrednosti kontrola). Ovo su asinhroni događaji.
- Alternativa je *polling* metoda programiranja, kada se u svakoj iteraciji, proveravaju vrednosti određenih kontrola i ako je došlo do promene izvršava se određeni deo koda. Mana je korišćenje resursa, koje je kod *event-driven* eliminisano, već se koriste ugrađeni mehanizmi operativnog sistema.
- *Event* struktura se nalazi na paleti *Programming » Structures*.
- *Event* struktura se, u većini slučajeva, smešta unutra petlje, jer bi se u suprotnom izvršila samo jednom.
- Osim aktivnosti korisnika, *event*-ovi se mogu definisati i za externe I/O uređaje. *Events* registruje operativni sistem (OS) ili LabVIEW.



Event in Loop

- Registrovane *event*-e "hvata" *Event* struktura i prenosi odgovarajućem *case*-u. Takođe, presnose se i informacije o *event*-u.
- U slučaju pojave više *event*-ova, prvi se odmah obrađuje, a ostali se smešaju u red i obrađuju prema redosledu pojavljivanja.
- Moguće je "zaključati" UI sve dok se ne obradi tekući *event* (*default* podešavanje) .
- Čekanje na *event* ne opterećuje CPU.
- *Event* struktura omogućava filtriranje *event*-ova, npr. odbacivanje *Panel Close event*-a.
- Moguće je definisati dinamičke *event*-ove. Standardne *event*-ove može registrovati samo VI sa kojim korisnik "komunicira", dok ih ostali VI (pozvani kao SubVI) "ne vide". Dinamičke *event* "vidi" i SubVI.

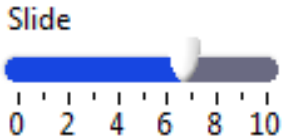


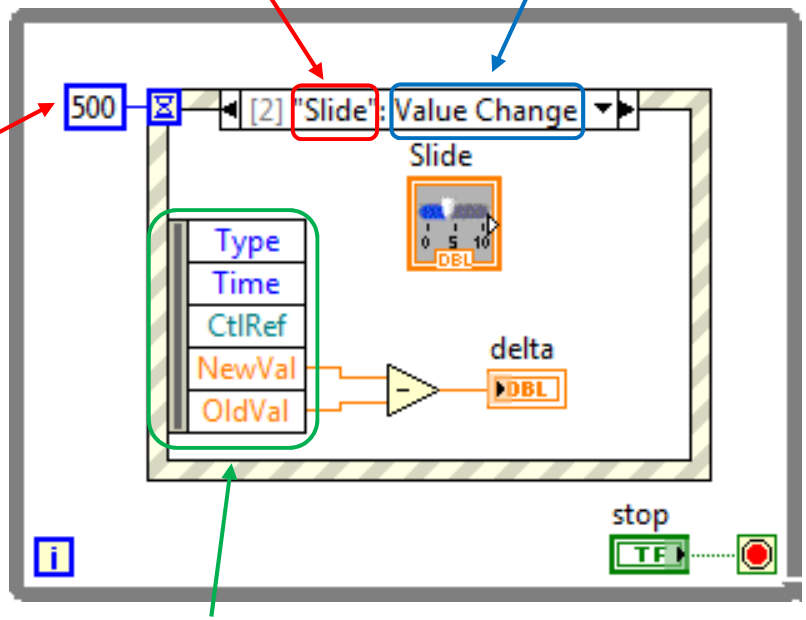


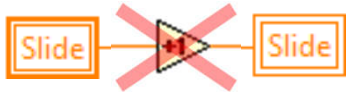
Event in Loop

Desni klik na struktura za dodavanje *Event*-ova.

Koliko *Event* struktura da čeka na *event* -1 za beskonačno.

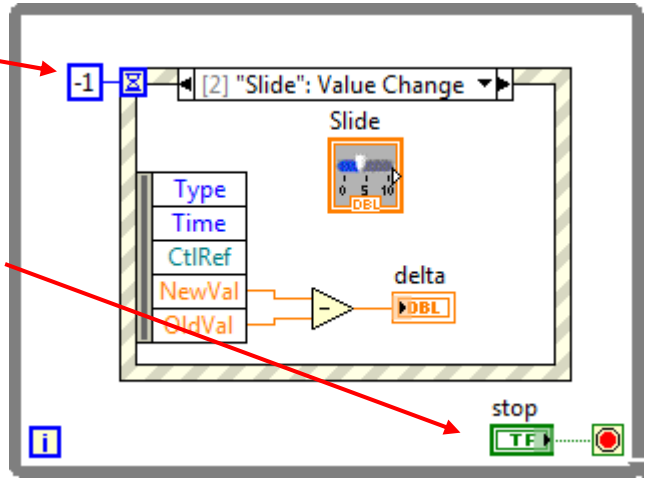






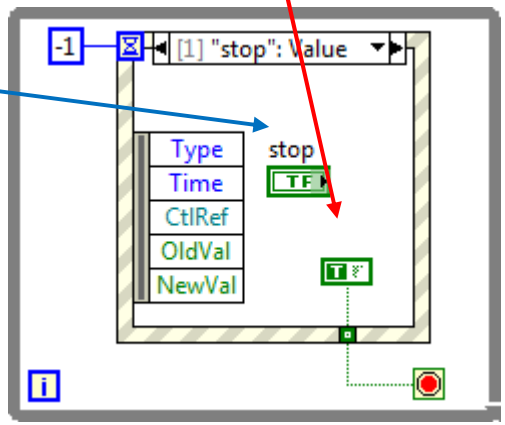
Izmena kontrole pomoću lokalne promenljive nije *Event Value Change*, jer je nije genirisao korisnik.

Petlja se vrlo teško prekida, jer **stop** mora da se postavi na ***true*** pre nego što izvrši neki od registrovanih *event*-ova.

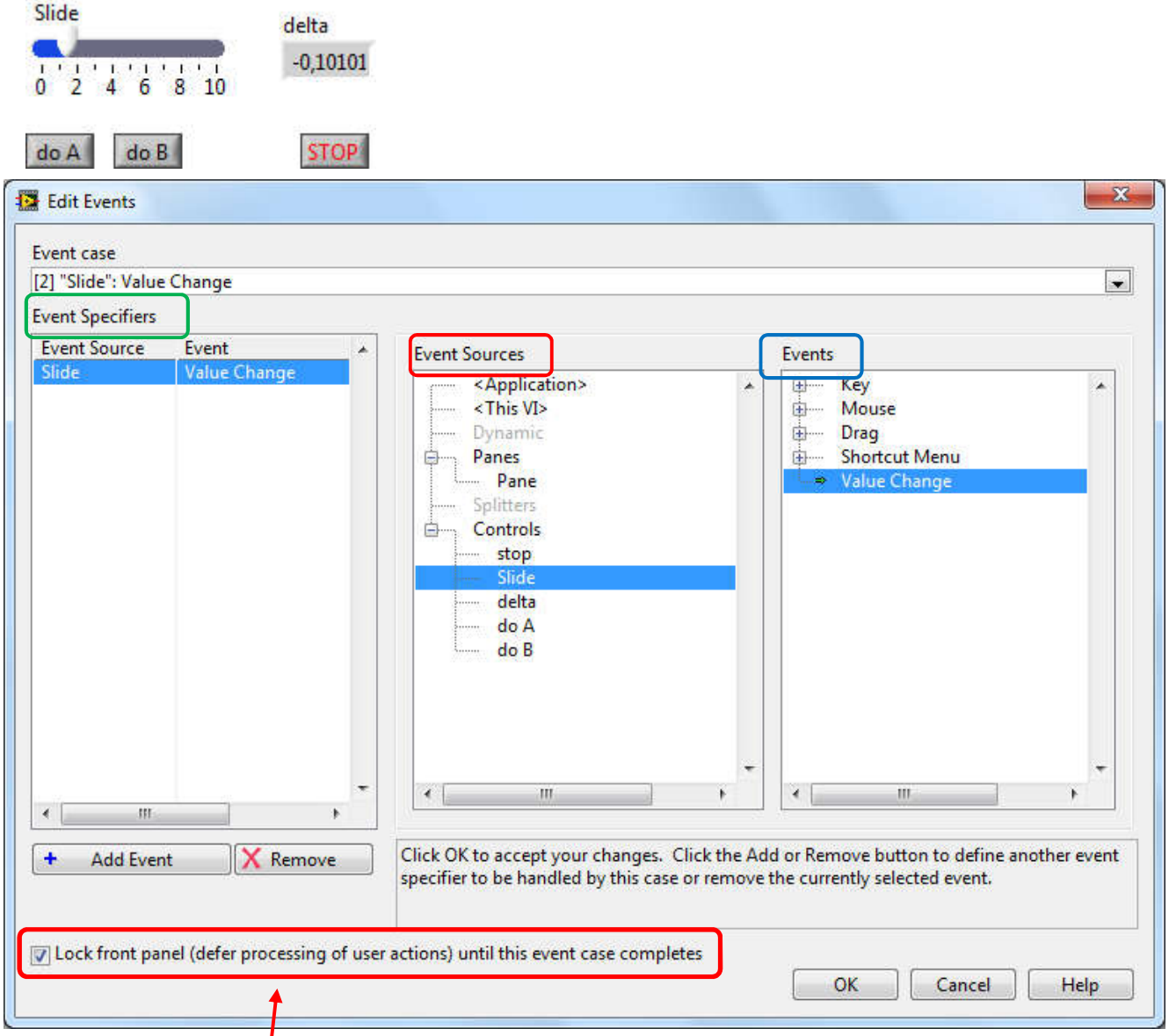


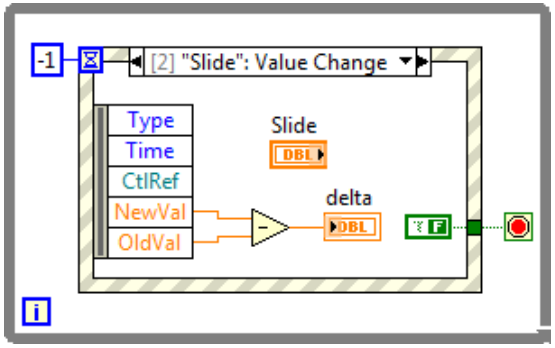
Latch mora biti "pročitan" u odgovarajućem *case*-u, ako želimo da mu se vrednost vrati na *false*.

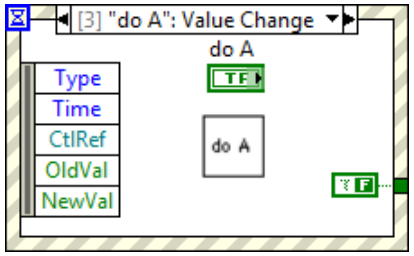
rešenje

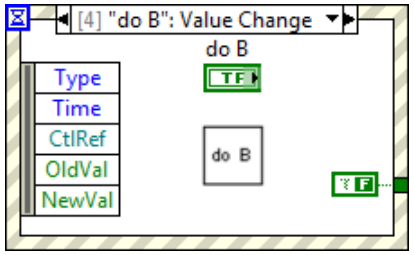


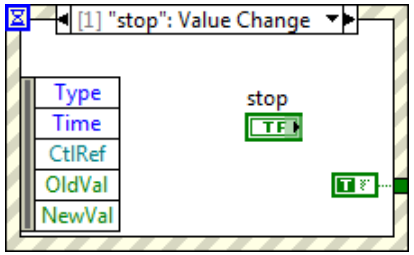
Event Structure





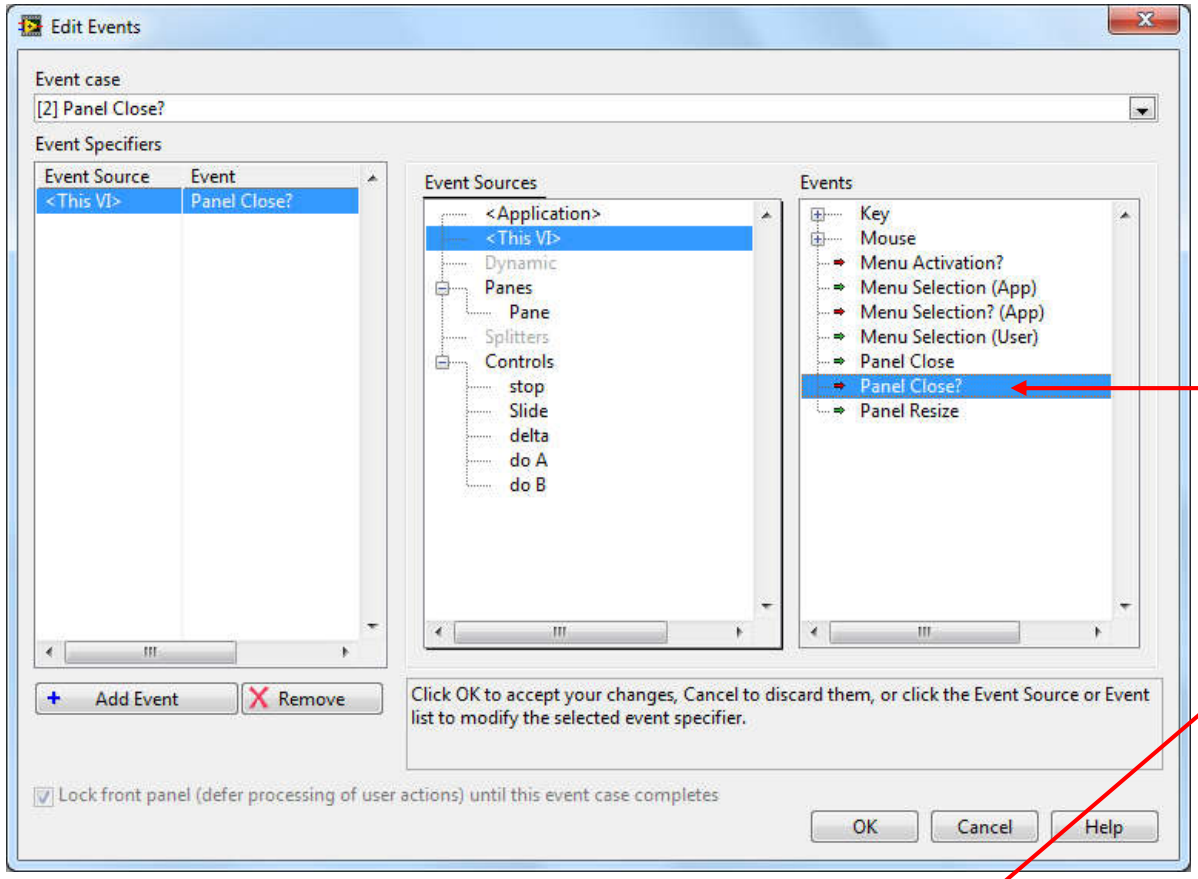






"Zaključavanje" FP-a (*default*).

Event Structure

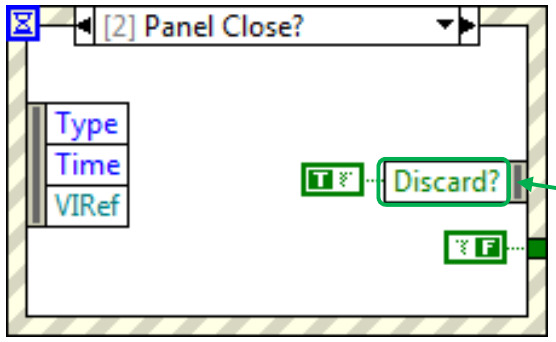


Standardno, *event* je *notify* tipa. (obaveštava da je korisnik već nešto uradio na FP).

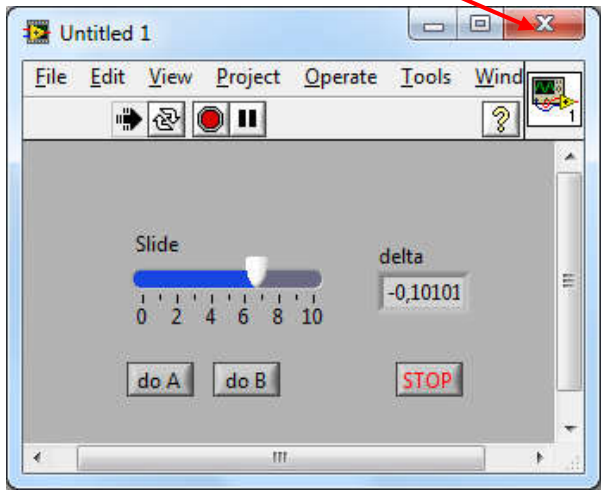
Postoji i *Filter event* ima crvenu strelicu i znak pitanja.

Filter event može otkazati akciju korisnika.

Sada ne reaguje na klik.



Event filter node

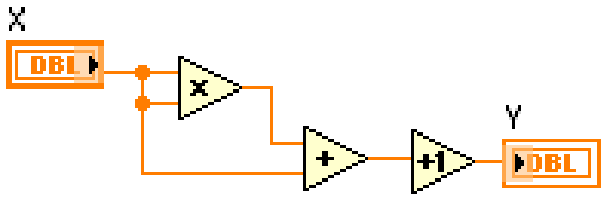


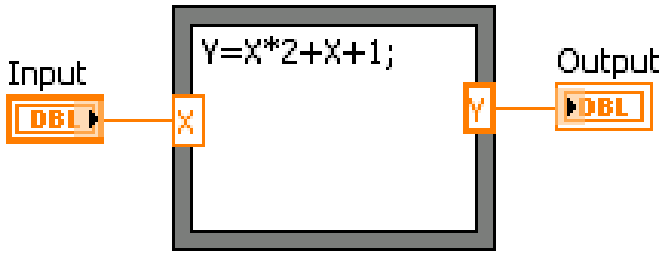
Vežba 12

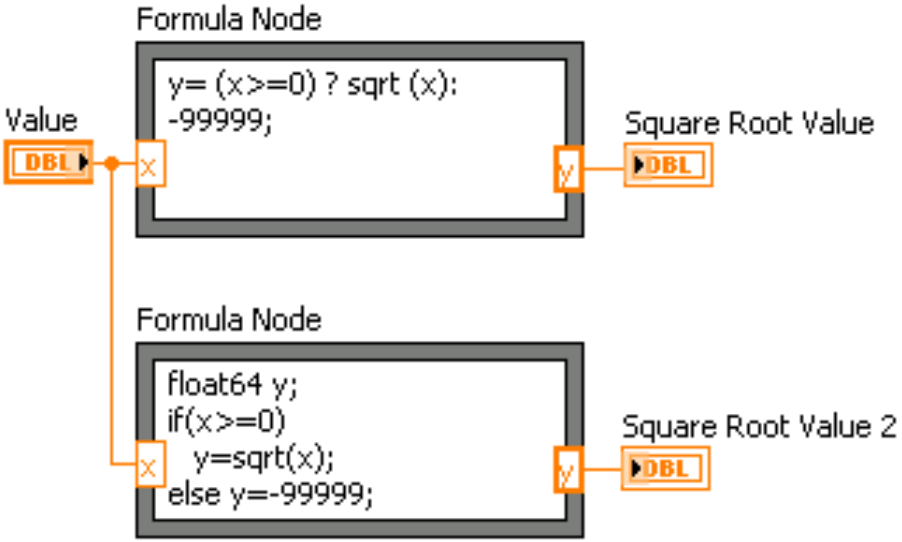
- Obezbediti da se prethodni primer (vežba 11) izvršava u *while* petlji pri čemu se koristiti *event* strukturu koja obezbeđuje sledeće:
  - ako je korisnik aktivirao stop taster, obaveštava ga o tome i napušta program,
  - ako je korisnik promeni neki od limita vrši proveru da li je gornji veći od donjeg i odbacuje promenu ako nije,
  - ako korisnik pokuša da zatvori panel postavlja pitanje da potvrdi i ako korisnik odgovori pozitivno zaustavlja se program.

Formula Node

- *Programming* » *Structures Palette*,
- Za izračunavanje komplikovanih jednačina,
- Promenljive se zadaju na ivicama,
- Imena promenljivih su *case sensitive*,
- Svaka naredba mora da se završi (;),
- *Context Help Window* prikazuje dostupne funkcije.



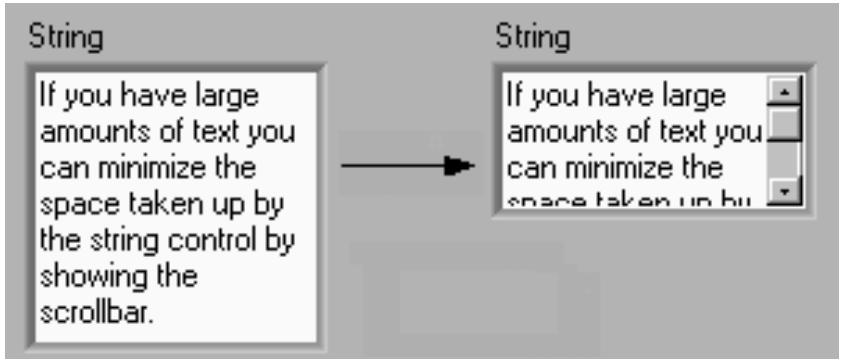




Oba *Formula Node* daju isti rezultat.

Stringovi

- *Controls» Modern » String & Paths*



Normal display

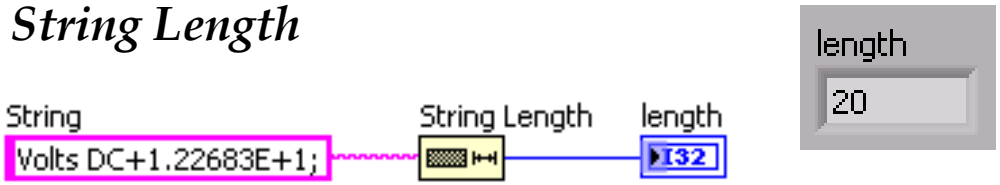
\ code display

Password display

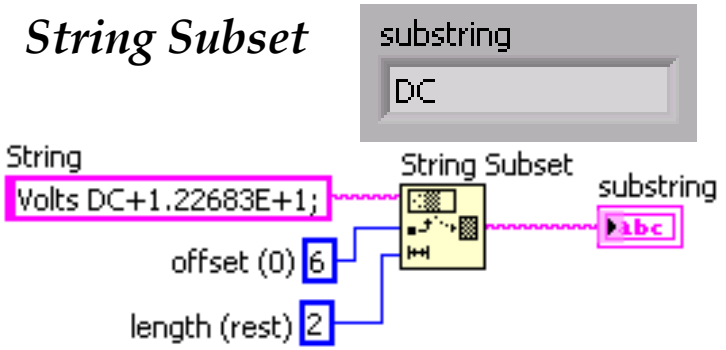
Hex display

Funkcije za rad sa stringovima

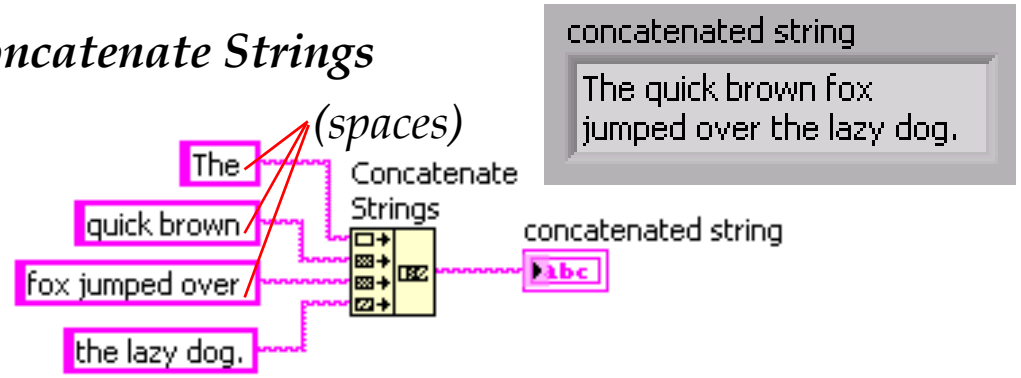
String Length



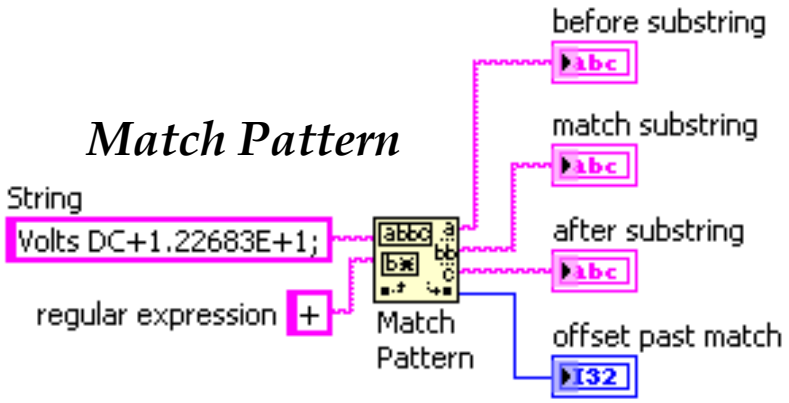
String Subset



Concatenate Strings

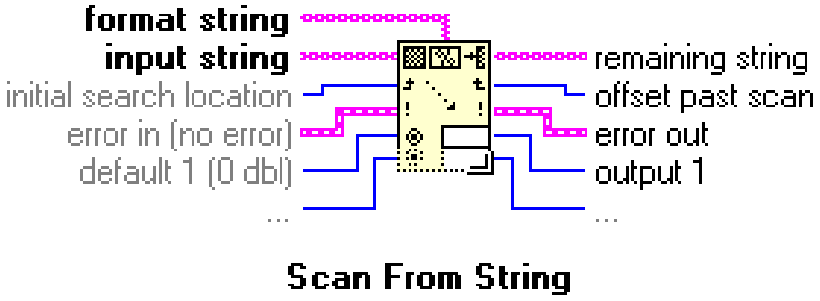


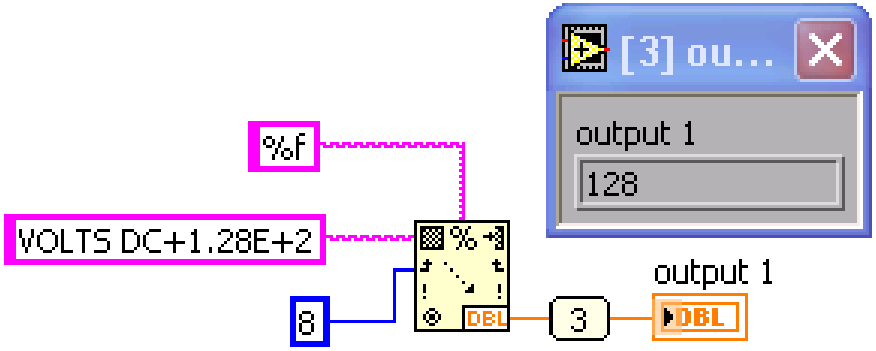
Match Pattern



Funkcije za rad sa stringovima

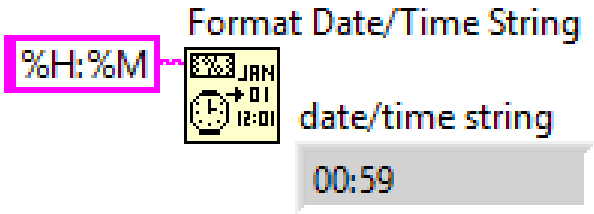
Scan From String

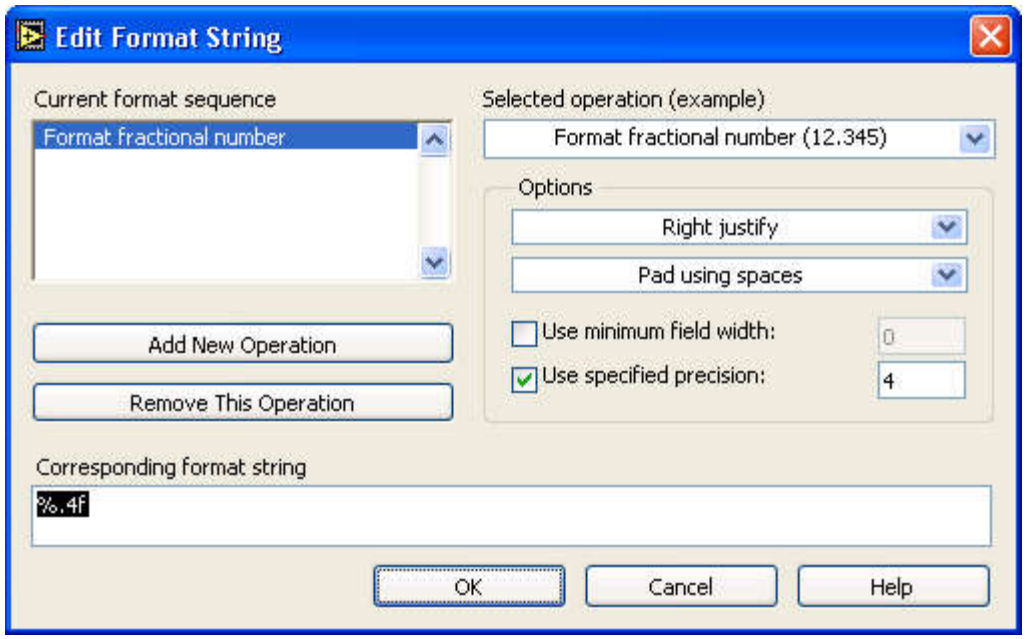


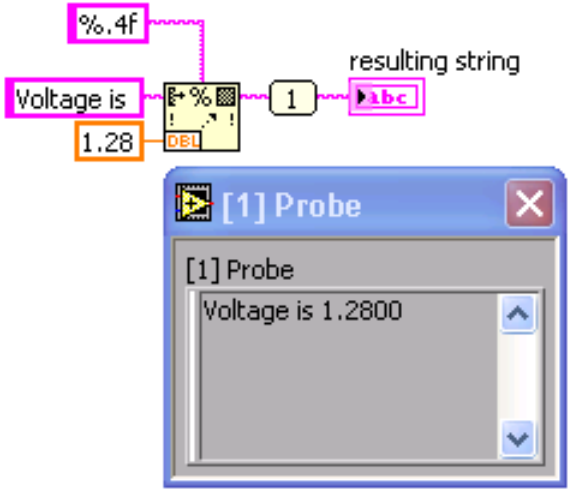


Format Into String

Formatiranje datuma

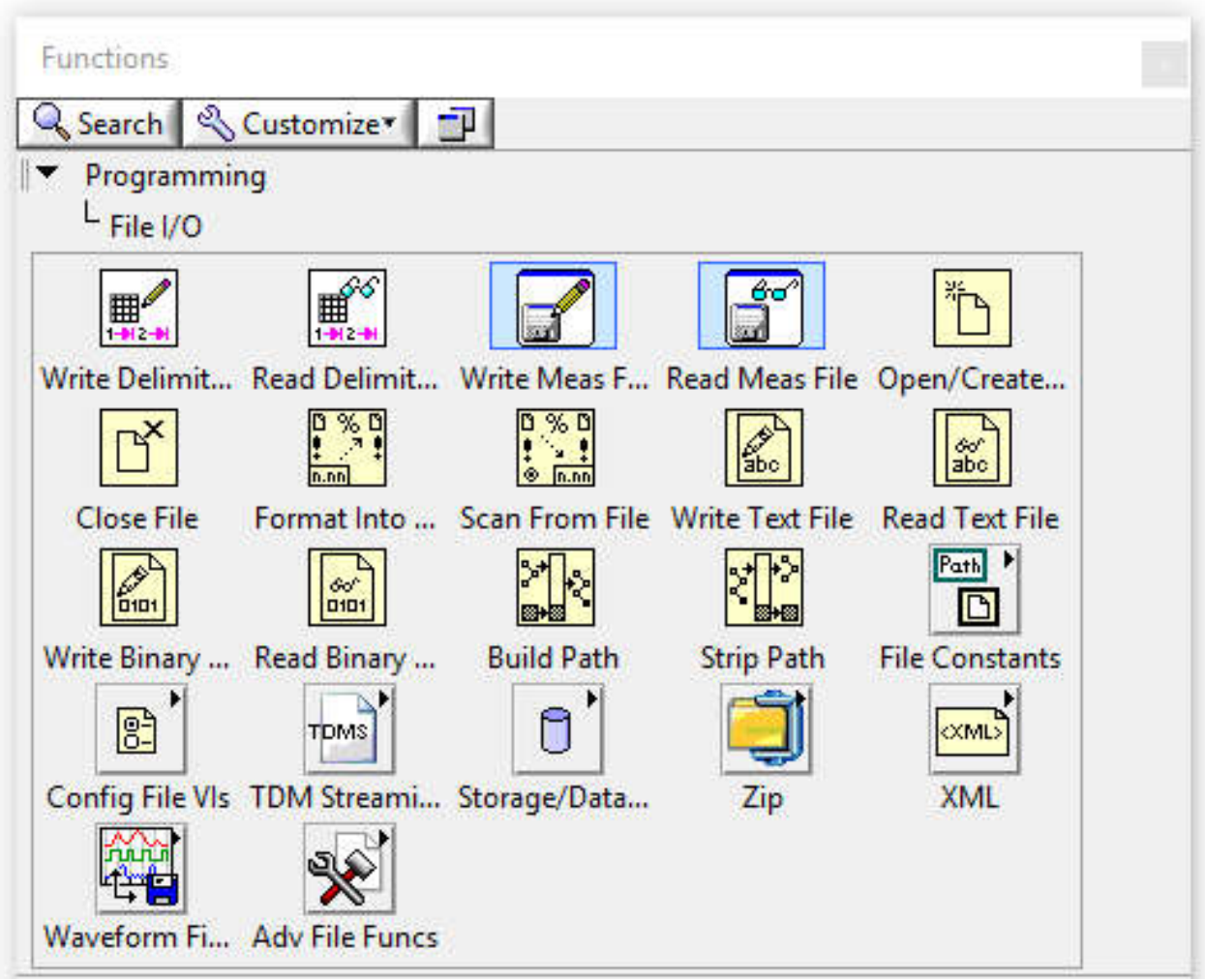






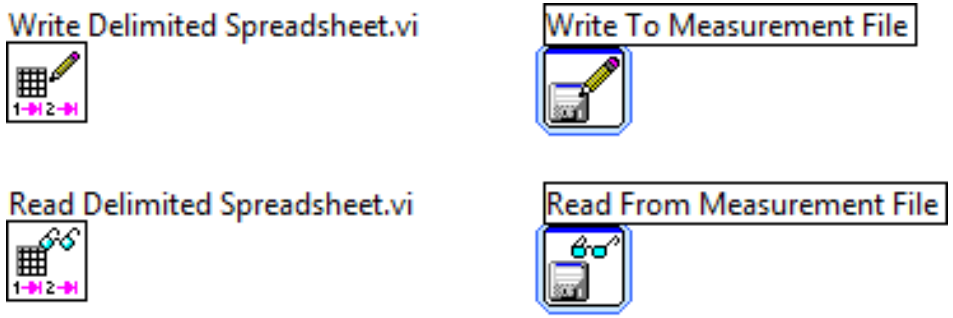
Rad sa fajlovima

- *Programming » File I/O Palette.*
- Veliki broj funkcija: *High Level VIs, Low Level VIs, Expess VIs, Config File VIs.*



Rad sa fajlovima

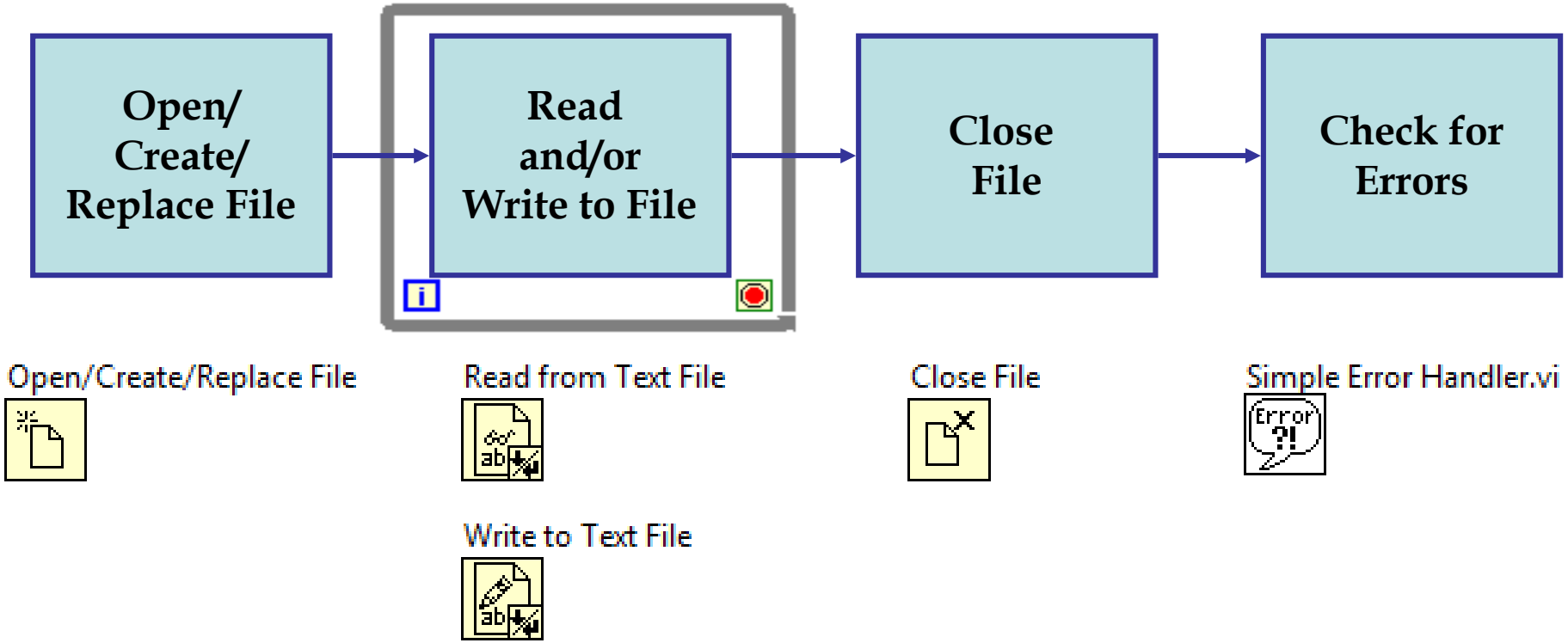
- Ukoliko se samo jednom upisuje u fajl mogu se koristi *High Level* ili *Express Vis*.



- Vežba 13: Kreirati tri niza od po 128 elemenata – sinusna funkcija, šum i *square* funkcija. Prikazati sva tri niza na istom *Waveform graph* kao i u tabeli, pri čemu svaki niz u po jednoj koloni. Snimiti sva tri niza u fajl korišćenjem funkcije *Write Delimited Spreadsheet*. Koristiti *Signal Generation Pallete*.

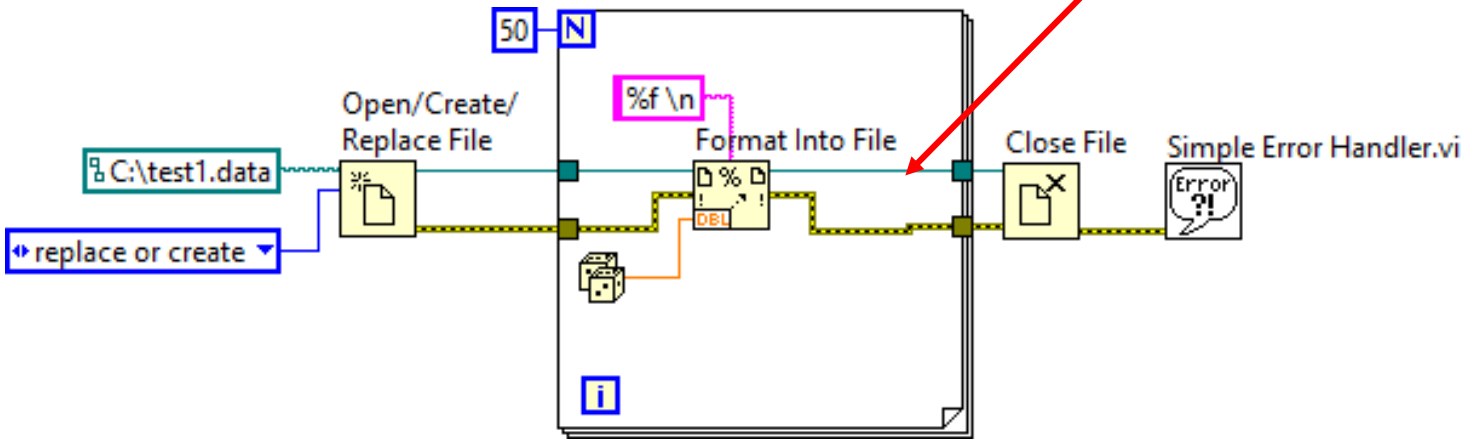
Rad sa fajlovima

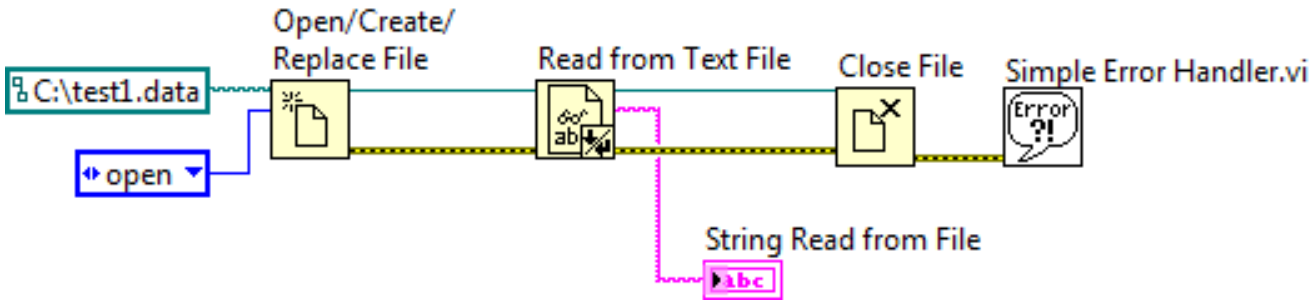
- Najčešći način rada je pomoću *Low Level VIs*.



Upisivanje u fajl i čitanje iz fajla

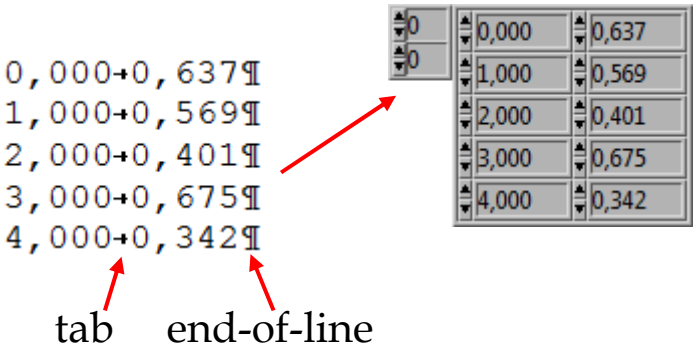
Refnum – jedinstveni identifikator fajla

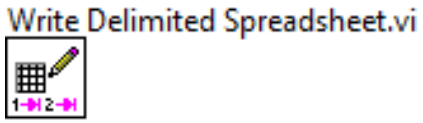


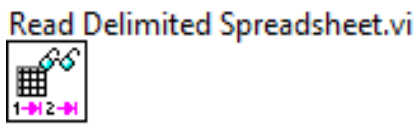


Spreadsheet fajl

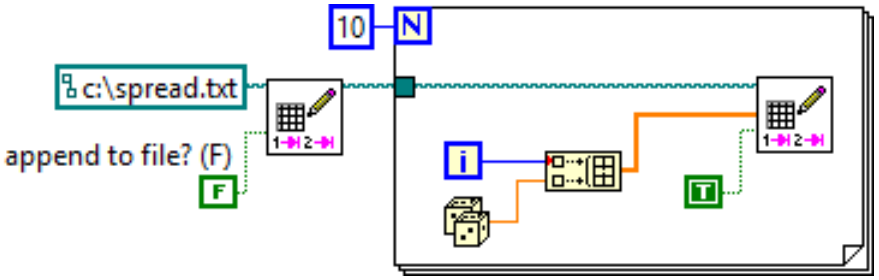
- *Spreadsheet* fajl je popularan alat za manipulaciju i analizu podataka.
- Postoji nekoliko formata spreadsheet fajl, a najzastupljeniji je *tab-delimited*:
  - Kolone su odvojene tab karakterom,
  - Redovi su odvojeni *end-of-line* karakterom.



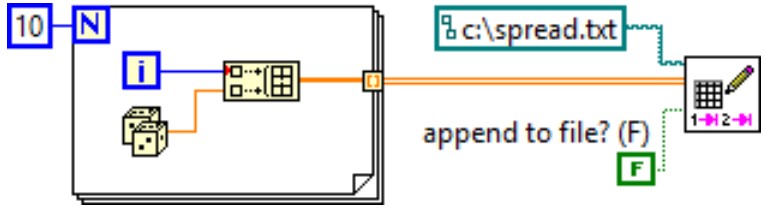


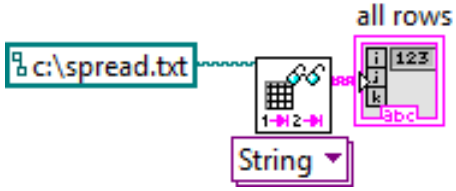


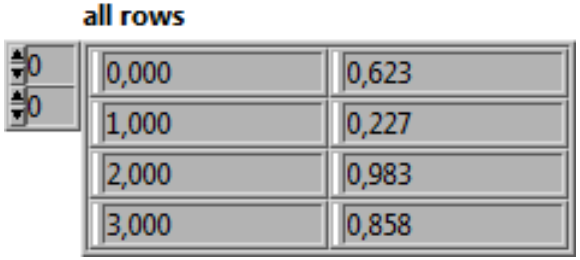
1D unos



2D unos





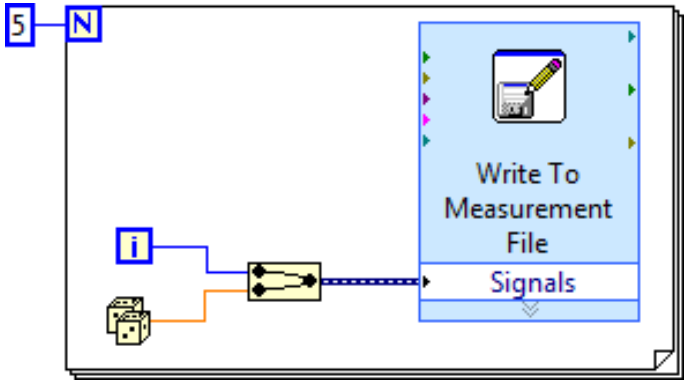


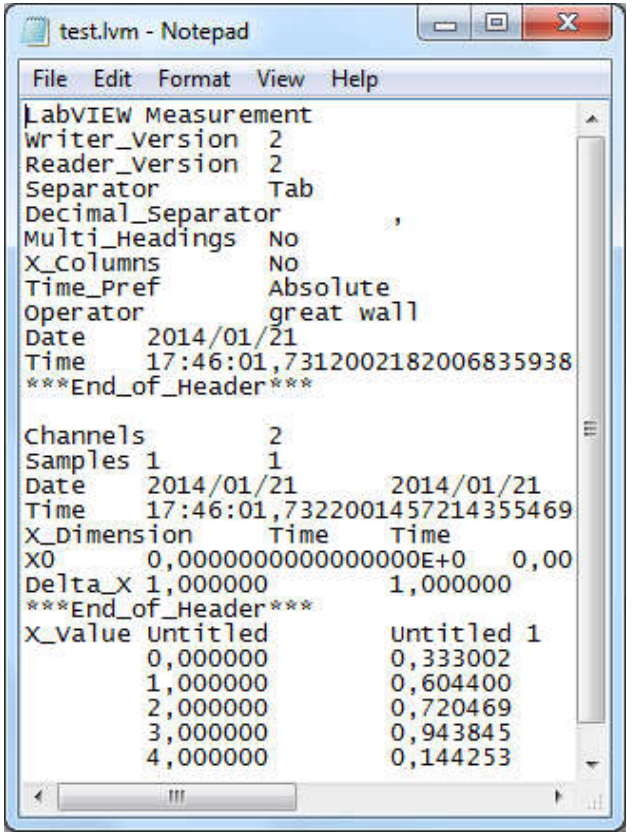
Vežba 13

- U toku izvršavanja petlje u svakom koraku upisivati slučajan broj u fajl. File je potrebno otvoriti samo jednom i zatvoriti samo jednom. Svaki slučajan broj se upisuje u jednom redu fajla u obliku "odbirak *i* je: *x*", gde je *i* redni broj odbirka, a *x* njegova vrednost.

LabVIEW *Measurement File* – LVM

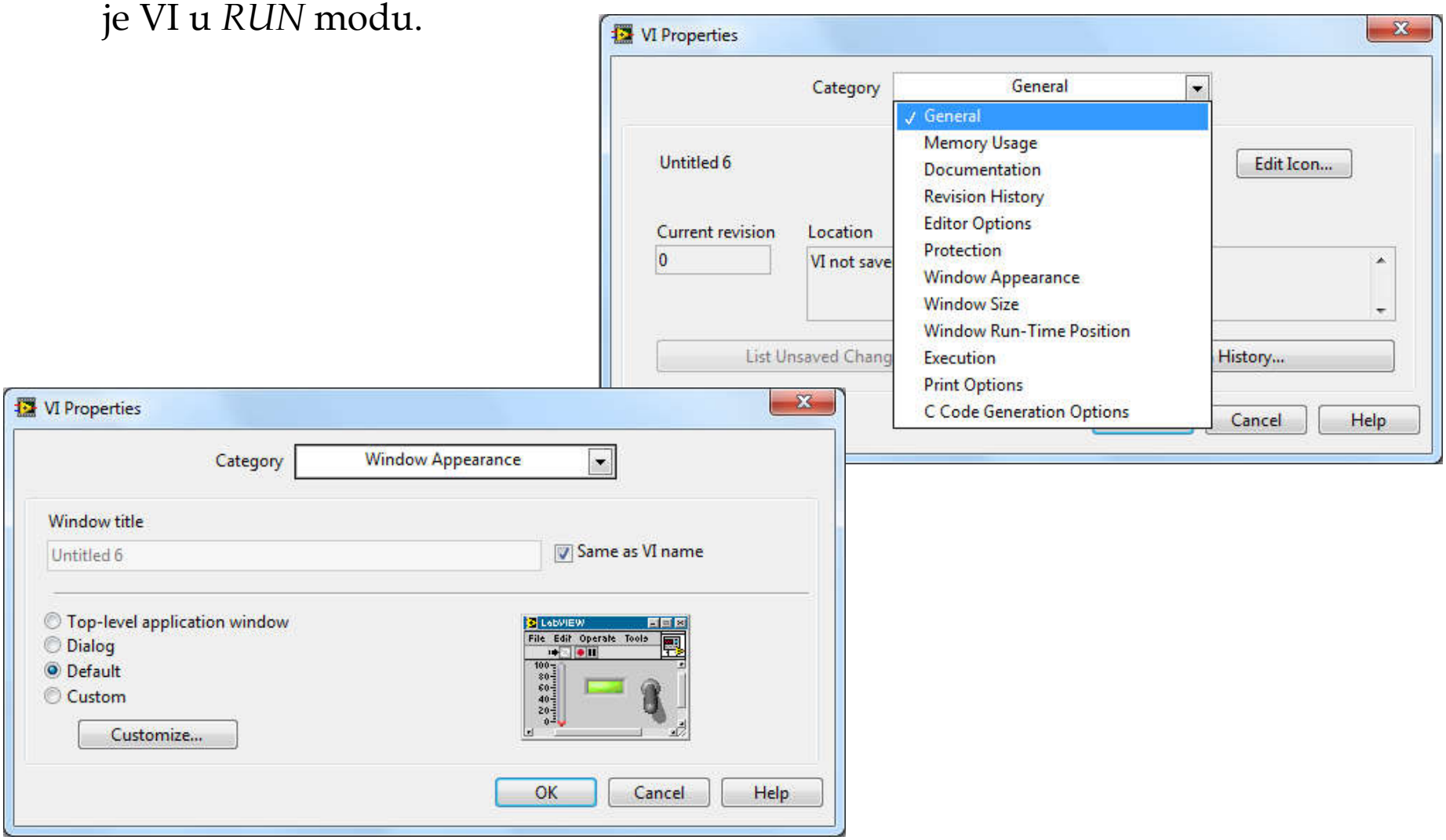
- Dve funkcije – Read/Write LabVIEW Measurement File.
- Uključuje otvaranje, čitanje/upis, zatvaranje i obadu greške u jednom.
- Podaci se formatiraju kao stringovi koji su ili *tab* ili *comma delimiter*.
- Mešoviti signali se čuvaju kao dinamički tip podata *dynamic data type*.
- Za veću kontrolu načina formatiranja podataka ili pisanje efikasnijeg koda, preporučljivije je koristi *Low-Level VIs*.





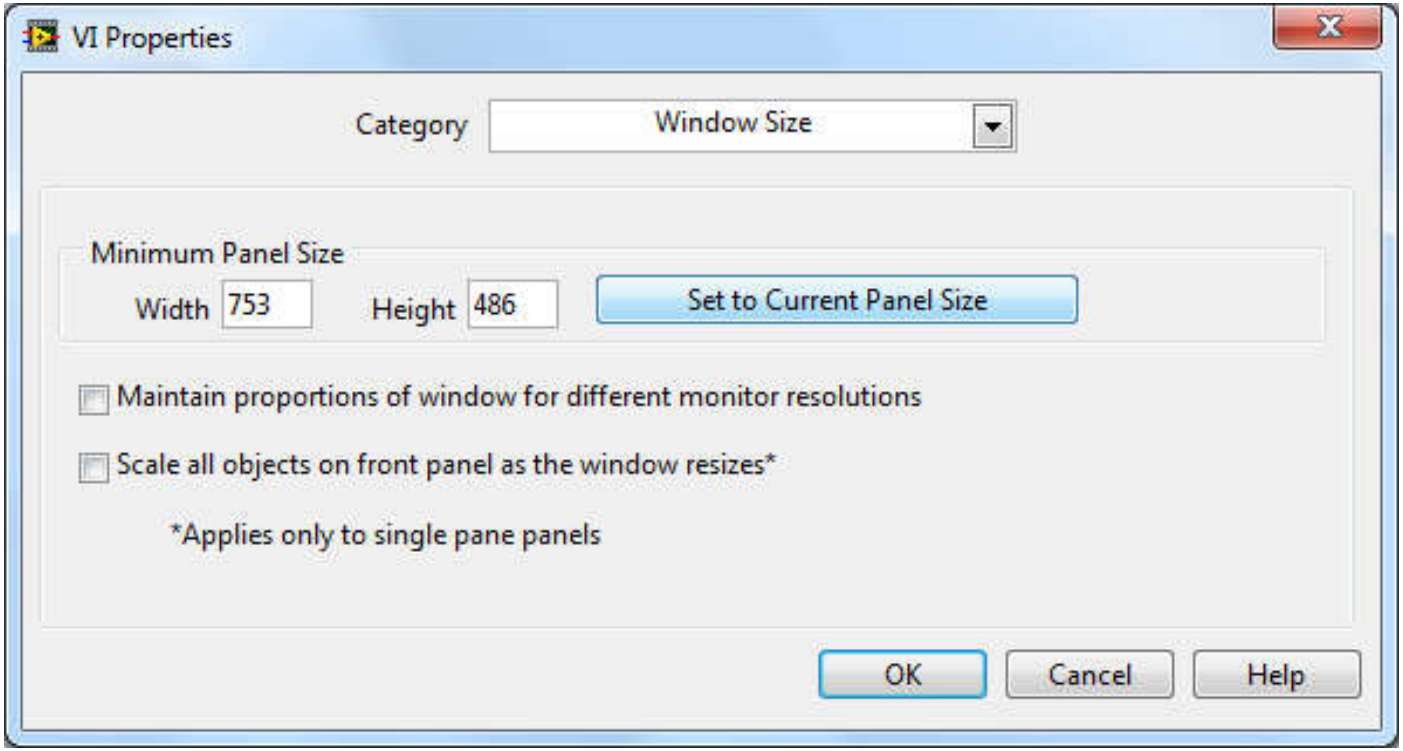
Customizing VI

- VI *Properties* su dostupni desnim-klikom na ikonu VI ili iz *File* menija.
- Promene se odnose na sve instance VI u svim aplikacijama i vidljive su kada je VI u *RUN* modu.



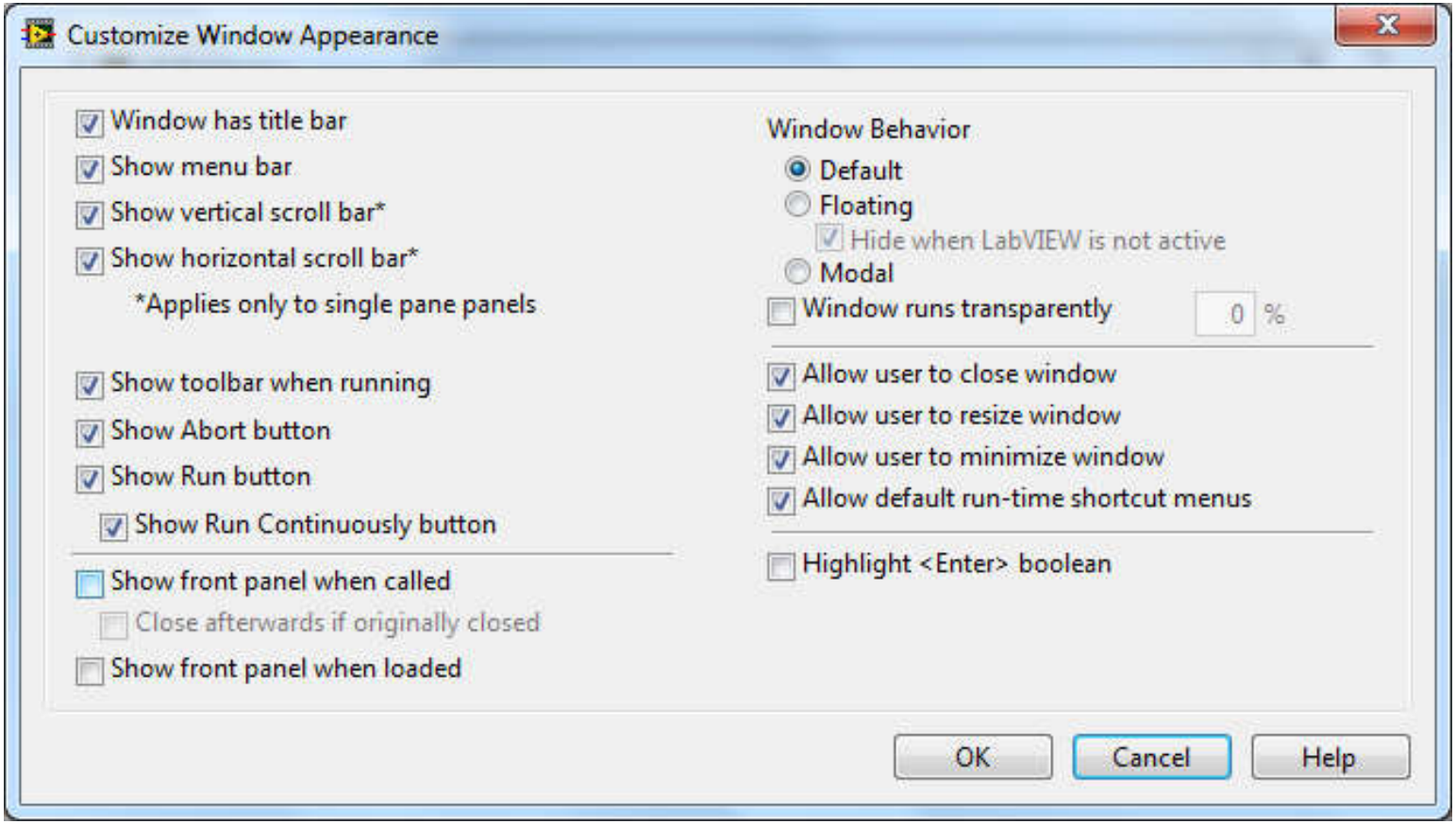
Customizing VI

- *Window size:*
  - *Set minimum and current panel size,*
  - *Adjust size of panel relative to the monitor,*
  - *Scale objects on panel as window resizes.*



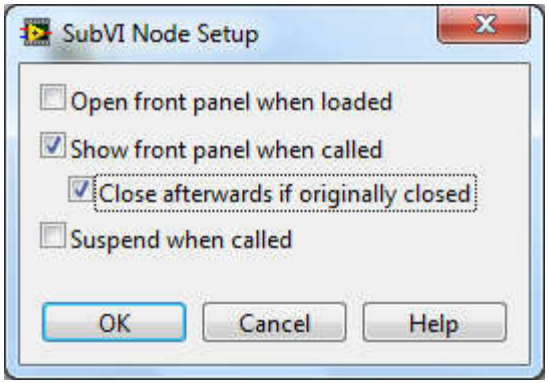
Customizing VI

- *Create custom window appearance.*

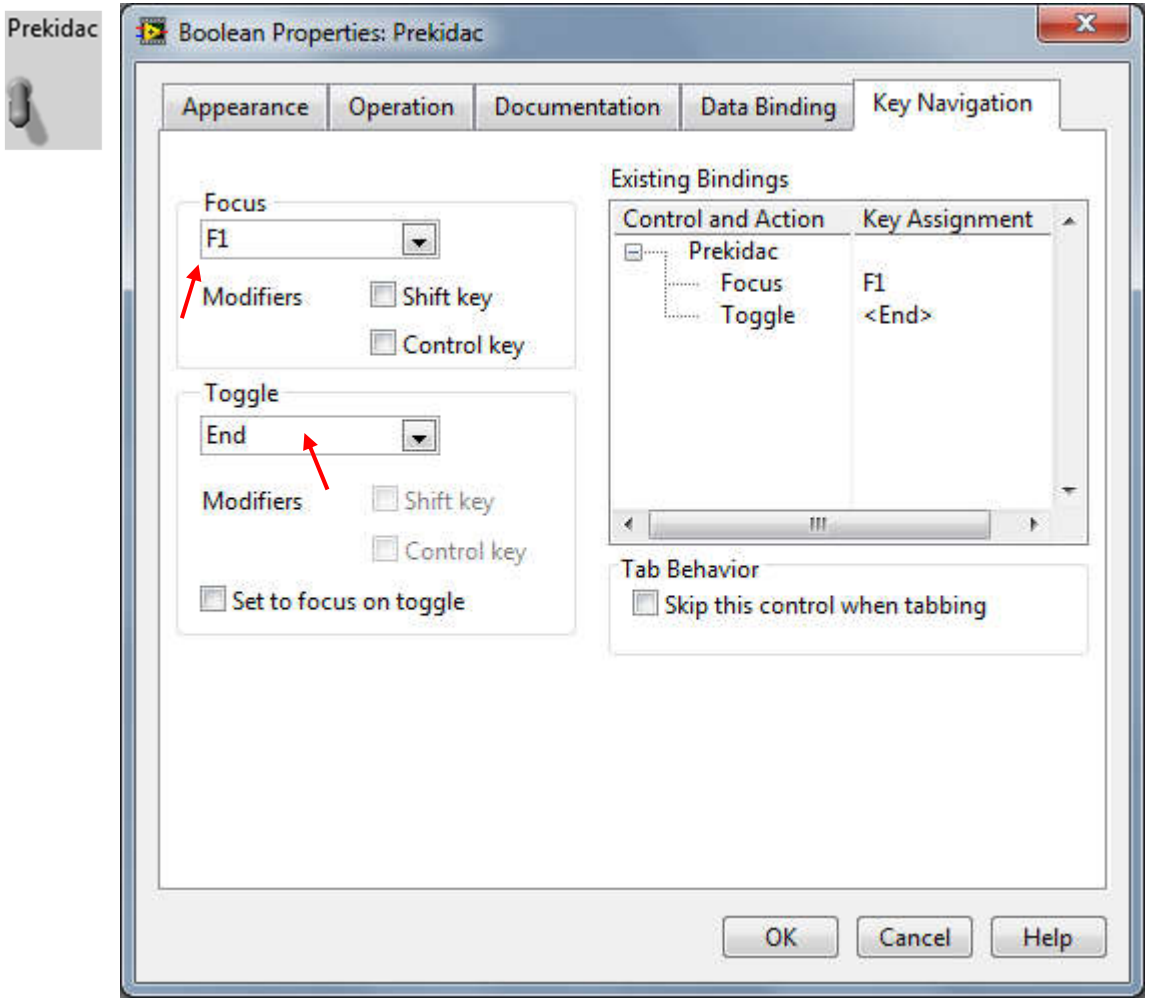


Customizing VI

Podešavanje SubVI prilikom poziva.

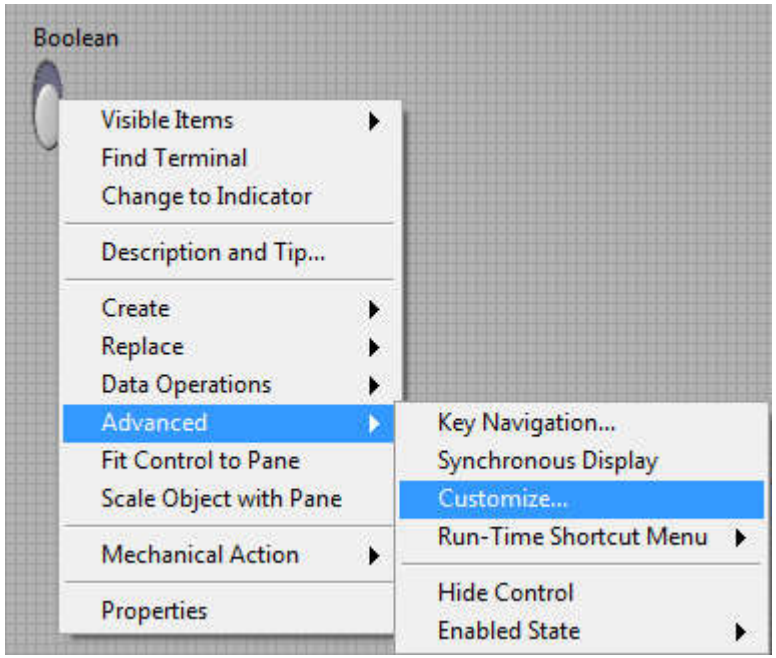


Definsanje pristupa terminalu pomoću tastature.

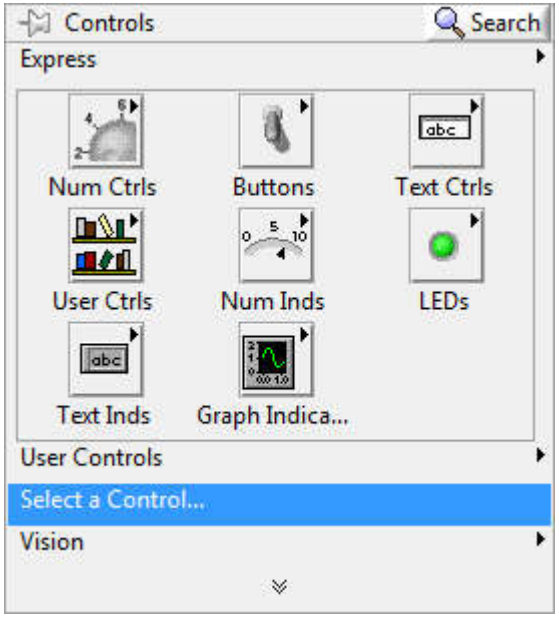


Custom Control

- Formiranje korisničke kontrole – poseban fajl, ekstenzija .ctl
- Može se koristiti u bilo kom VI.

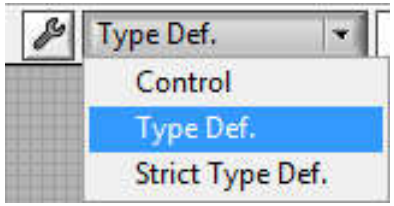


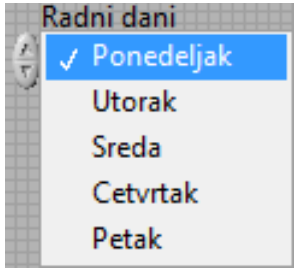
- Postavljanje na FP-u.



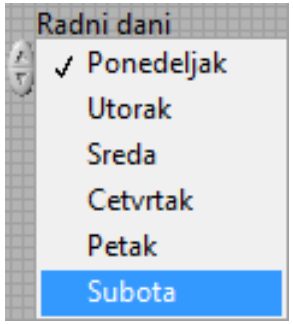
Type Definition i *Strict Type Definition*

- Pri promeni *Custom Control*-e svi njene instance koje su već definisate (postavljene na FP nekog VI) se ne menjaju.
- *Type Definition* – sve promene koje su vezane za tip podataka kontrole se primenjuju na sve instance te kontrole.

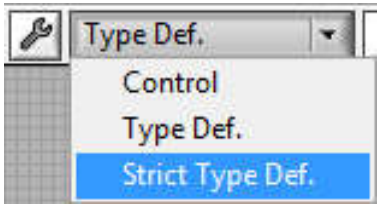


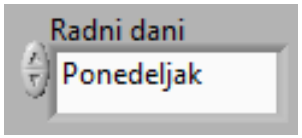


Dodat još jedan element za Enum tip podaka.

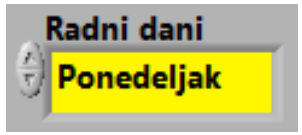


- *Strict Type Definition* – sve promene se primenjuju na sve instance te kontrole.



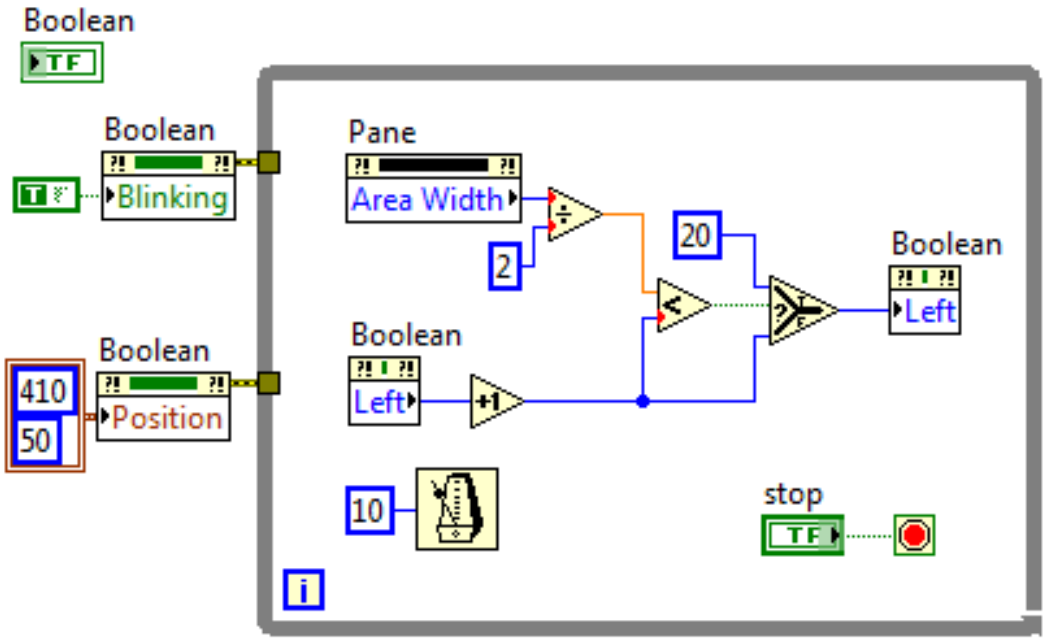


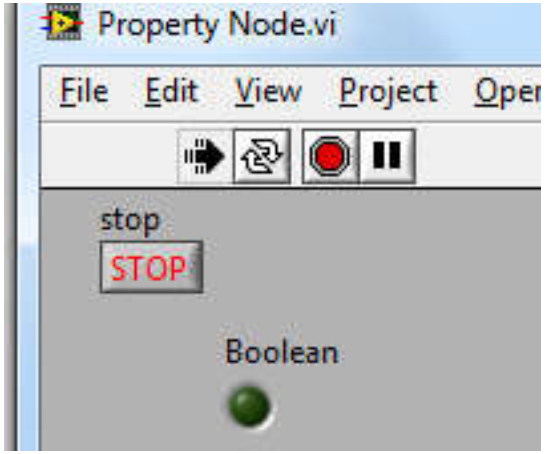
Promenjen font i boja pozadine.



Property Node

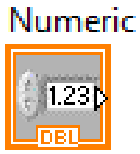
- Omogućava izmenu izgleda FP u toku izvršavanja programa, npr. određenim korisnicima je dozvoljen pristum samo nekim kontrolama, ostale moraju biti u stanju *Disabled*.
- Desni klik na terminal (BD) » *create* » *property node*, a zatim izbor *property-a* kojem se želi pristupiti.
- Ne podržavaju svi property funkciju *write*, već samo *read*.
- *Property Node* može biti povezan i sa delom FP-om u koji se mogu smestiti kontrole/indikatori (*Pane*) .

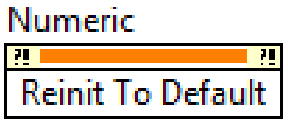


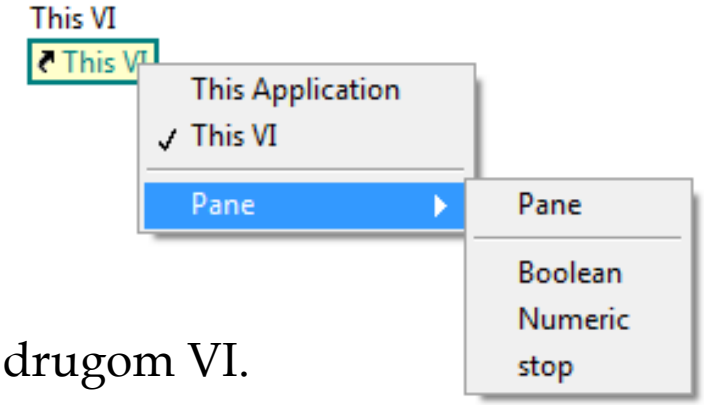


Invoke Node i *Control Reference*

- Koristi se za izvršavanje neke akcije na objektu u toku izvršavanja programa.
- Za razliku od *property* koji predstavlja neku o osobina obekata, *invoke node* je metod, tj. operacija koja se može izršiti nad objektom.
- *Invoke Node* se pristupa desnim klikom » *create* » *invoke node* i izbor metode koja se želi izvršiti.



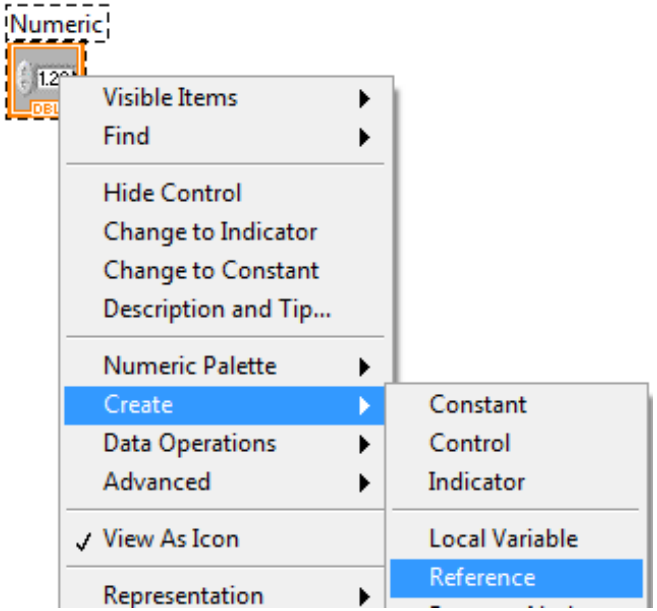




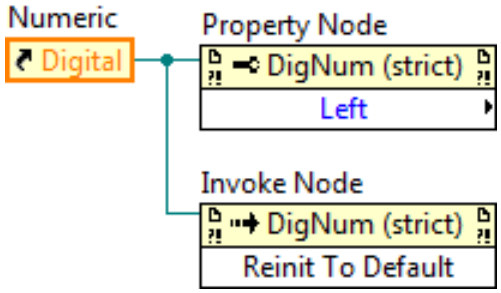
- *Control Reference* – pokazivač na objekat LabVIEW-a.
- Omogućava i referenciranje objekata koji se nalaze u drugom VI.
- Svaki objekat (indikator ili kontrola) ima jedinstveni identifikator (referencu) u LabVIEW-u. Referenca objekta postaje aktivna kada VI (a time i svi SubVI koji se poziva) na kome se objekat nalazi započne sa izvršavanjem.
- Može se još referencirati i aplikacija (LabVIEW), tekući VI (*This VI*) i sam panel (Pane) tekućeg VI.
- Paleta *Programming* » *Application Control* i izbor *VI Server Reference*. Zatim klik na Referencu i moguće je izbrati objekat sa kojim će se povezati referenca.

Property Node, Invode Node i *Control Reference*

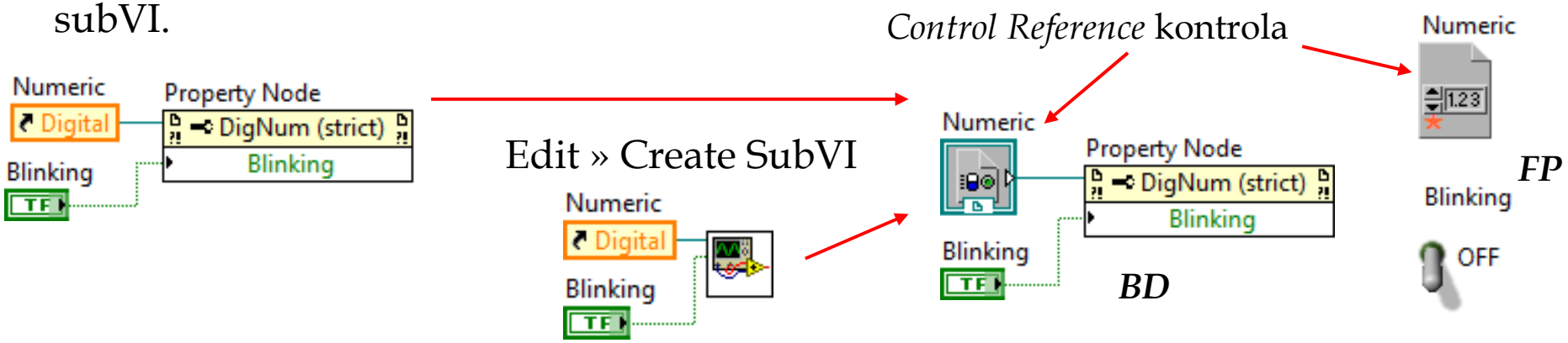
- *Control Reference* za objekat FP-a se može dobiti i desnim klikom i izborom *Reference*.



- *Control Reference* omogućava pristum *Property*-ima i *Method*-ama odgovarajućeg objekta pomoću *Property Node* i *Invoke Node* iz palete *Programming » Application Control*.



- *Control Reference* omogućava izmenu *Property*-a terminala iz glavnog VI-a unutar subVI.



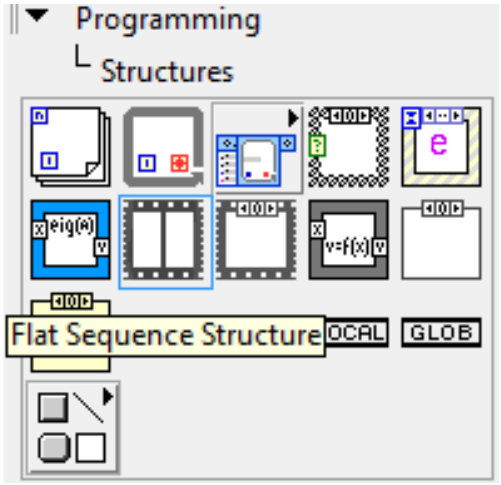
Sada se promena *Property*-a vrši u SubVI-u

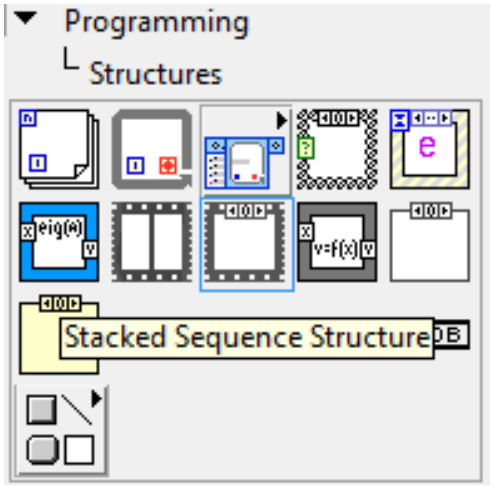
Vežba 14

- Nastavak vežbe 12: omogućiti promenu boje dijagrama pomoću Enum kontrole. Promena boje se mora izvršiti u posebnom subVI. Napomena: dodati još jedno stanje.

Sequence strukture

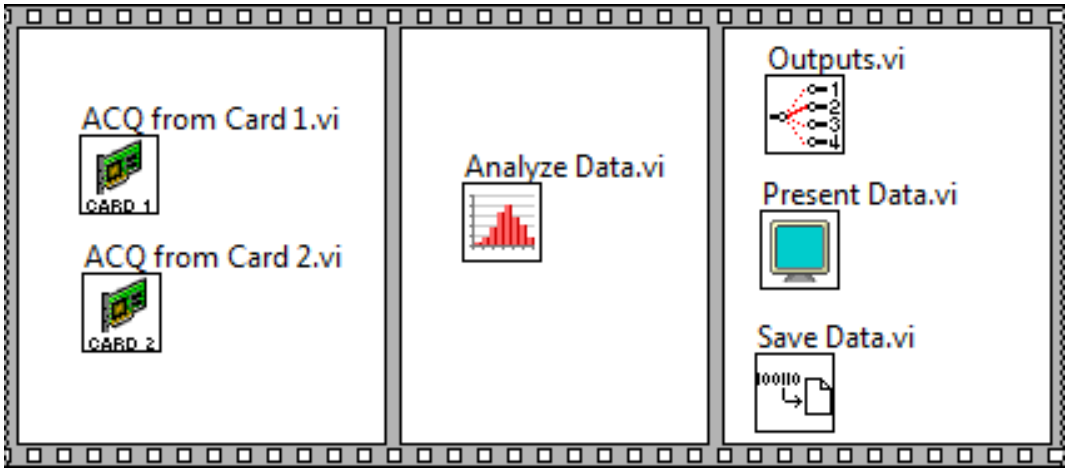
- Za razliku od klasični programski jezika LabVIEW ne izvršava programski kod naredbu po narednu već se koristi *dataflow* model.
- U situacijama kada se deo koda mora izvršiti pre nekog drugog dela može se forsirati serijsko izvršavanje koristeći *Flat Sequence* ili *Stacked Sequence* strukture.



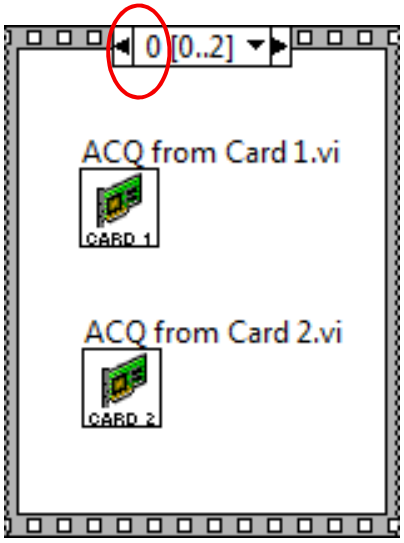


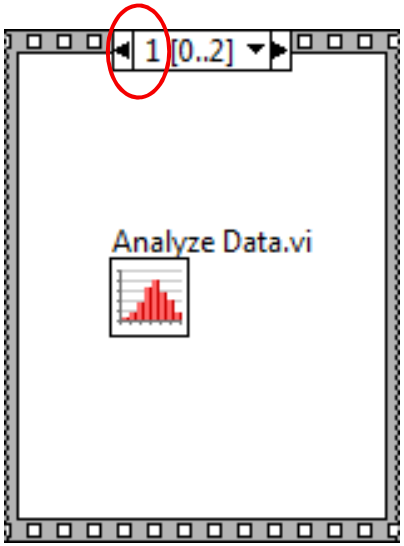
Sequence strukture

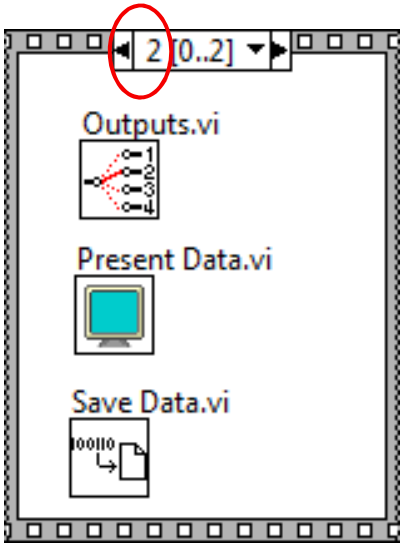
Flat Sequence



Stacked Sequence

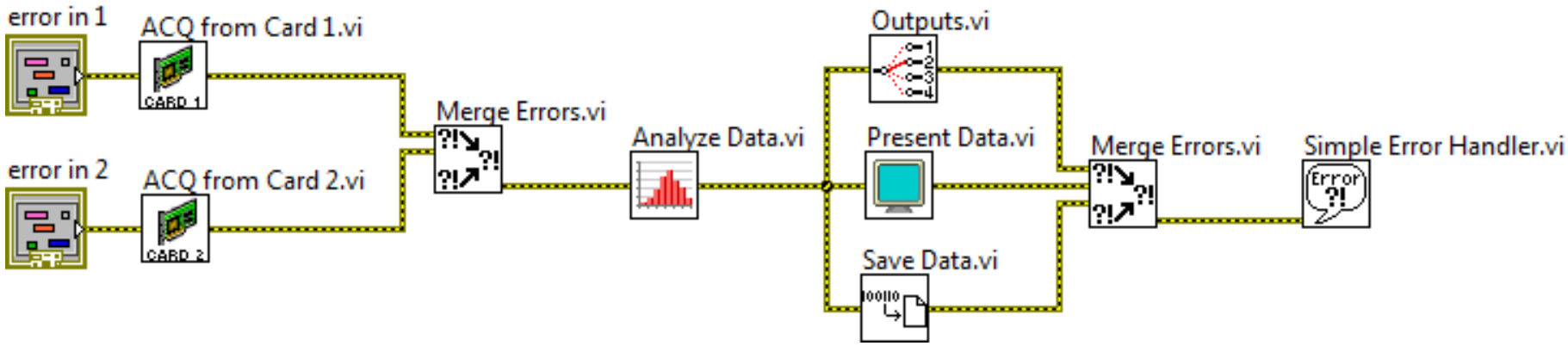




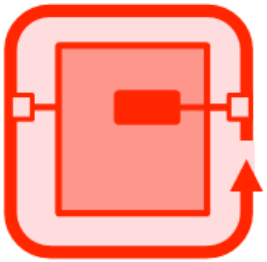


Sequence strukture

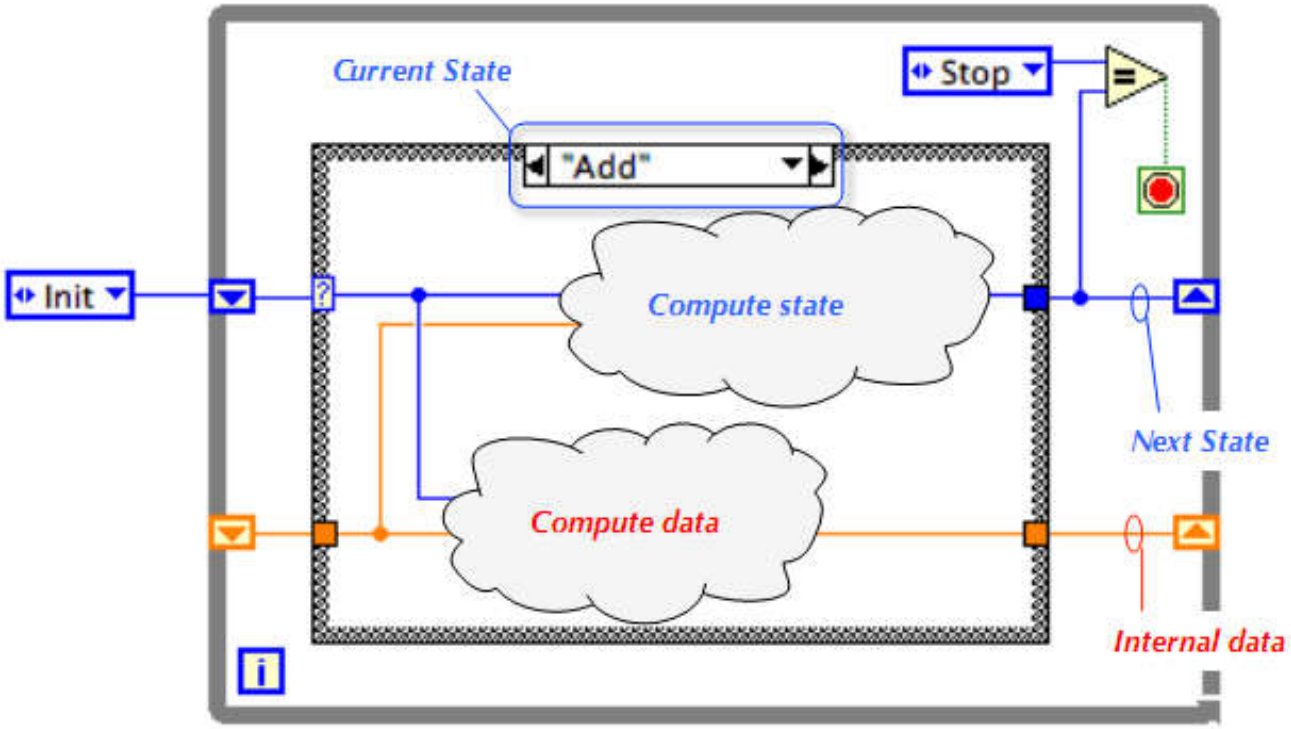
- Korišćenje *sequence* struktura se ne preporučuje jer se narušava *dataflow* model programiranja na koji se oslanja sam LabVIEW.
- Kao rešenje predlaže se, kad god je to moguće, umesto forsiranja serijskog izvršavanja korišćenjem sekvenciji, dodavanje *error in* i *error out* terminala za svaki subVI. Zatim, adekvatnim povezivanje *error in* i *error out* terminala moguće je odrediti tačan redosled izvršavanja subVI a ne narušti *dataflow* model programiranja.



Design Patterns – State Machine

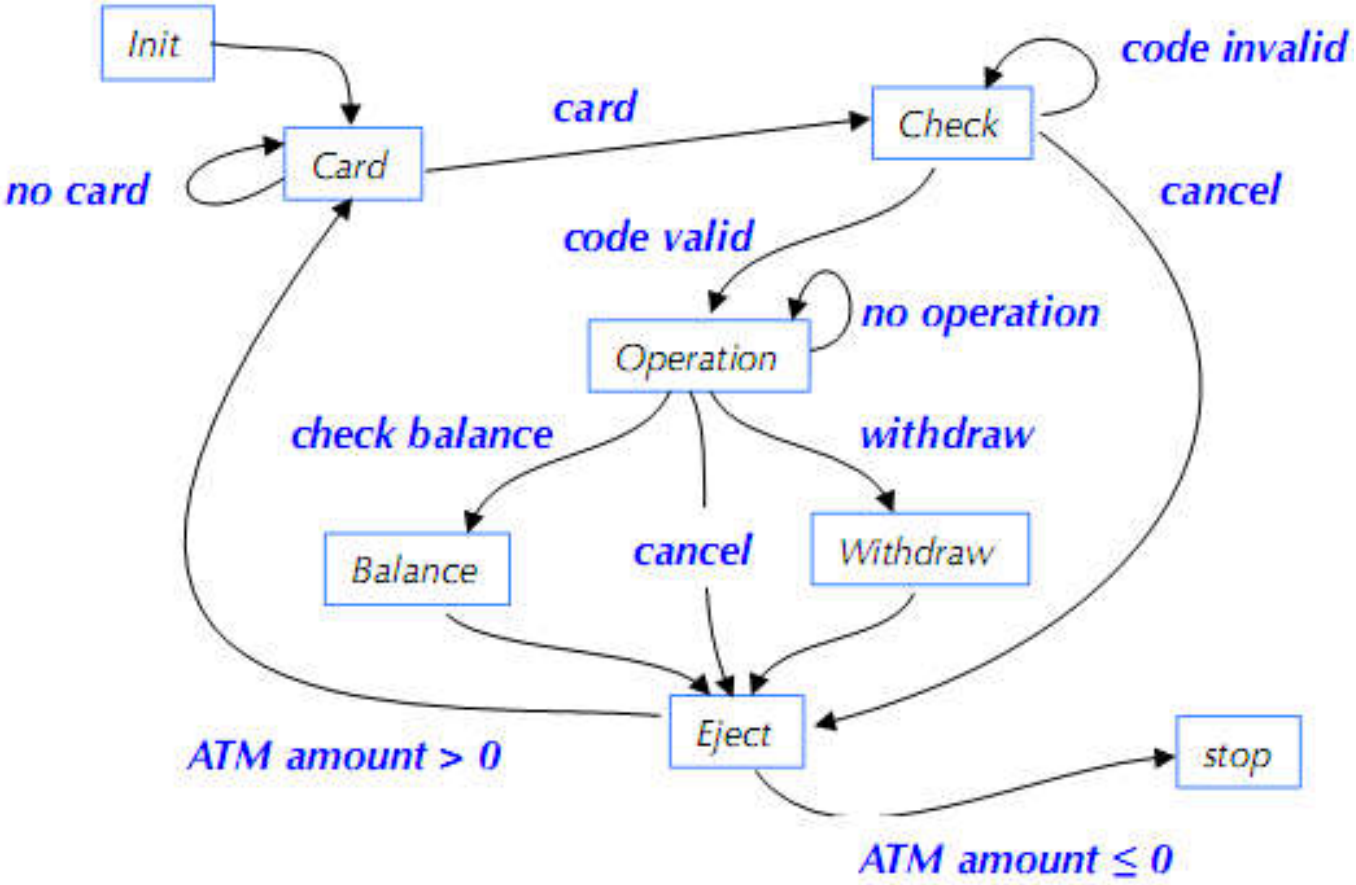


- *State Machine* – mašina stanja.
- Izvršava se sekvenca naredni, ali u odnosu na standardnu sekvencu, redosled koraka se određuje programski.
- Za razliku od *Sequence* može se prekinuti u proizvoljnom trenutku.
- Za promenu stanja koristi se *Shift Register*.
- *Case* struktura se koristi za svako stanje.
- U svakom stanju određuje se sledeće stanje.

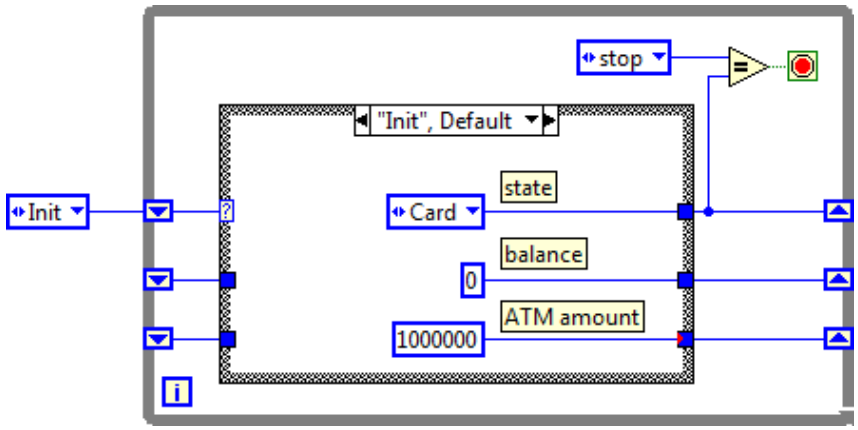


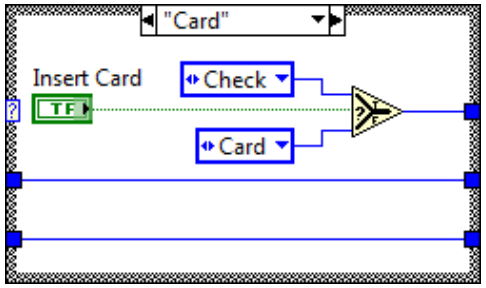

```
State = Init;
While (State!=Stop) {
  switch state {
    case Add: …
    case Sub: …
  }
  State = …
}
```

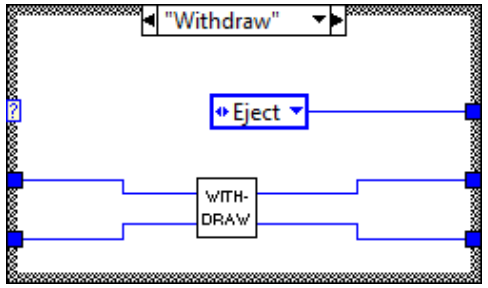
- Primer mašine stanja - ATM

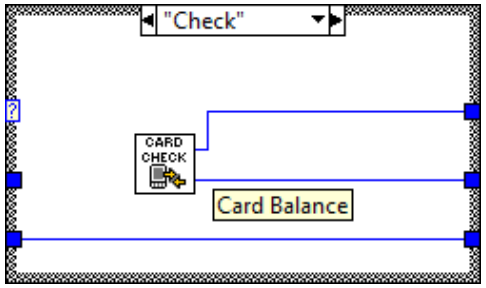


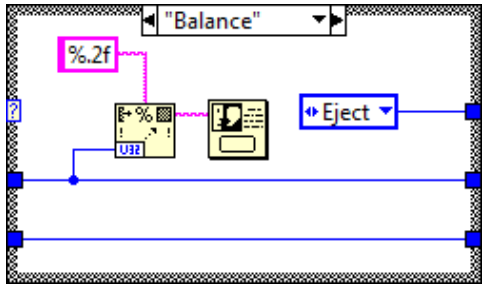
Design Patterns – State Machine

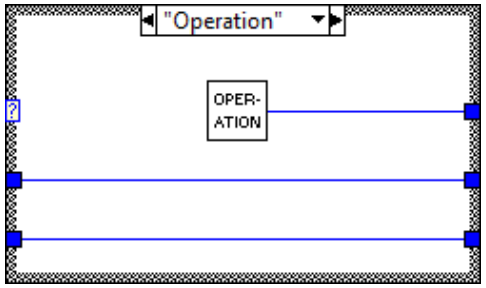


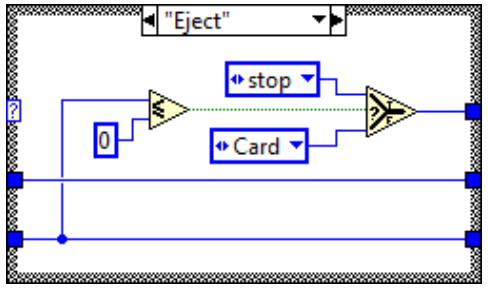


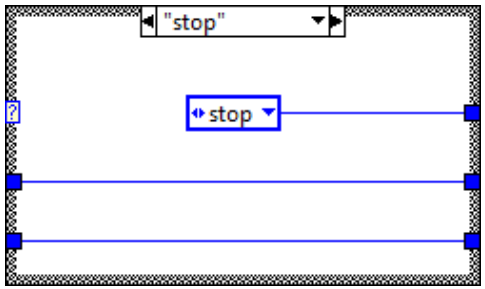






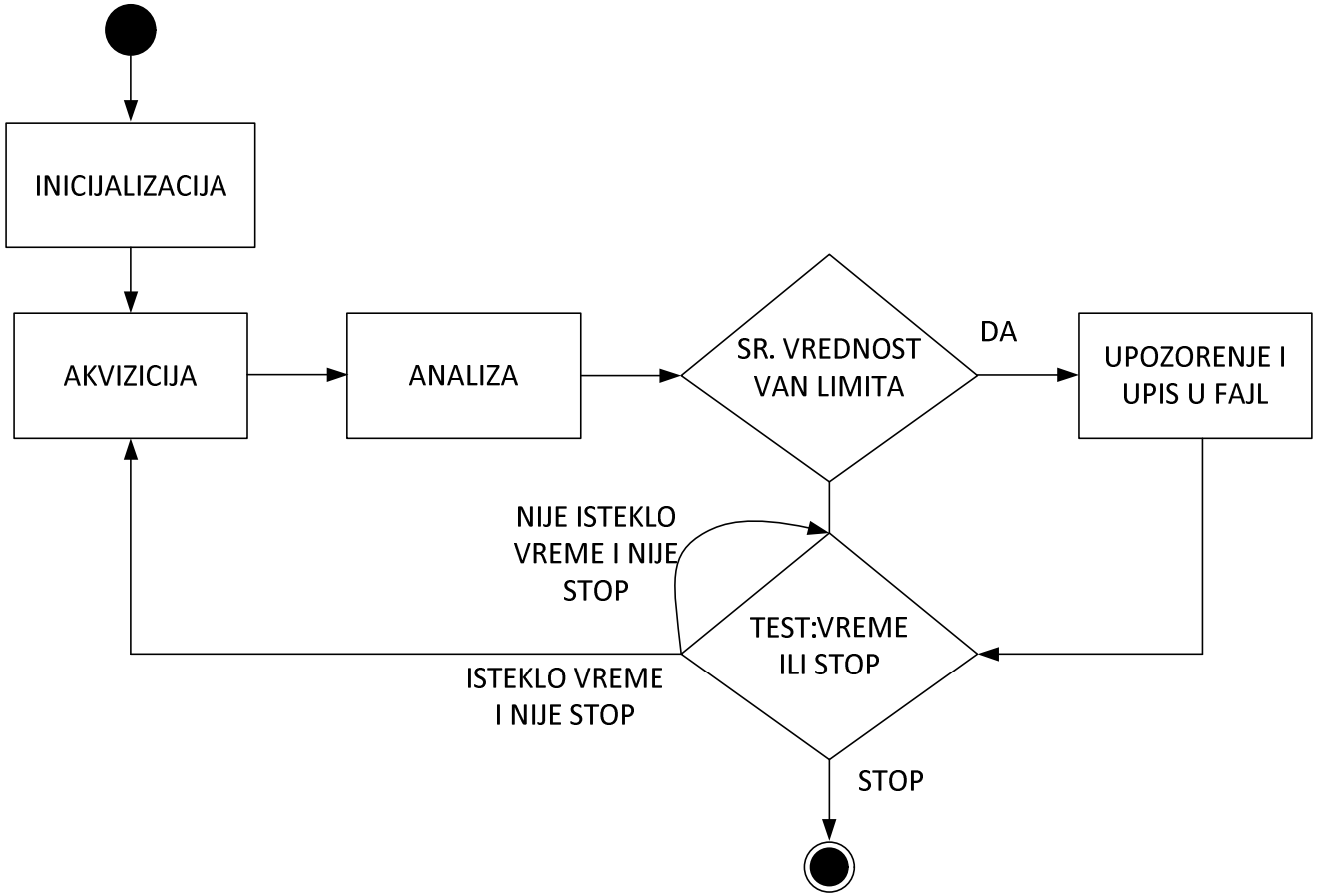






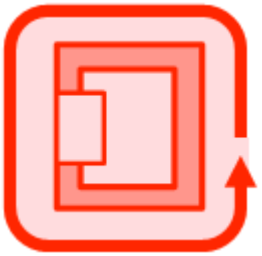
Vežba 16

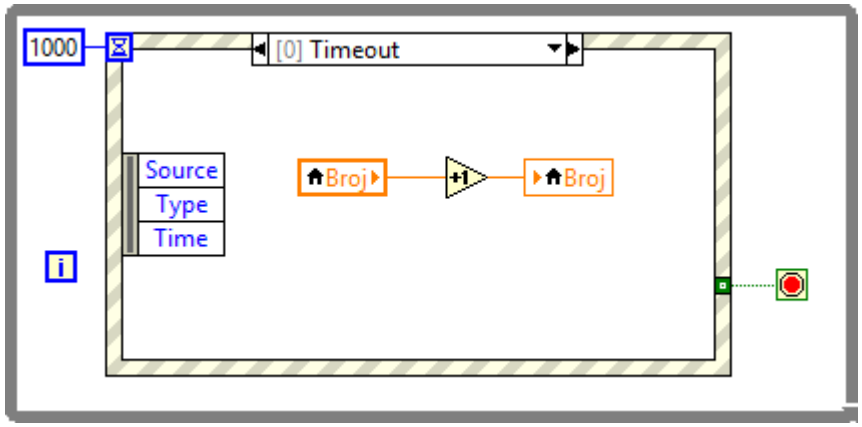
- Realizovati VI koji izvršava mašinu stanja na slici. U stanju inicijalizacije vrši se definisanje parametara simuliranog signala. U stanju analize određuje se srednja vrednost signala. Upozorenje se javlja ukoliko je srednja vrednost van dozvoljenog intervala. U fajl se upisuje vreme i datum upozorenja, srednja vrednost signala i limiti (gornji i donji). Fajl se otvara samo na početku i zatvara samo na kraju izvršavanja programa.

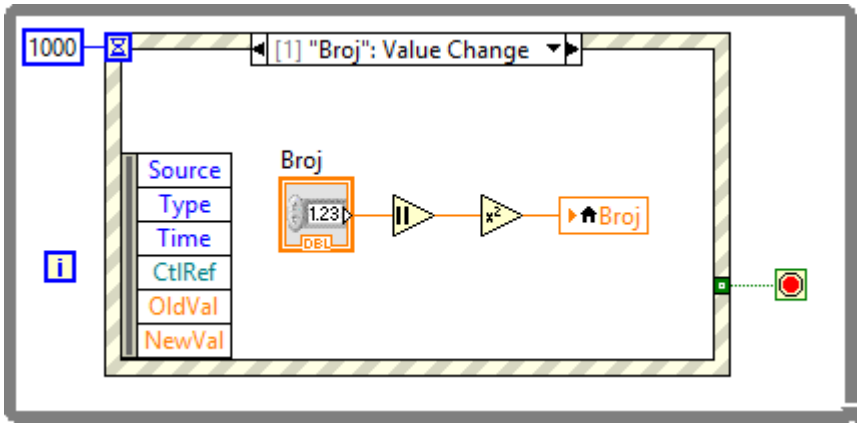


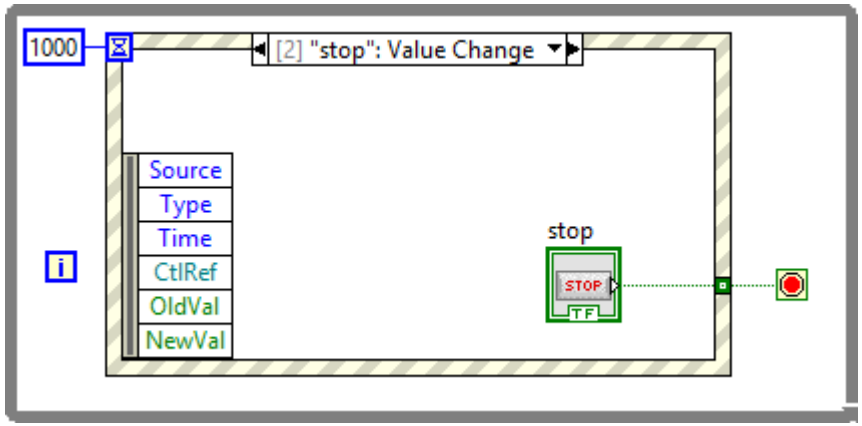
Design Patters – Event Loop

- Predstavlja standardno korišćenje *Event* strukture.
- Postoji nekoliko registrovanih *event*-ova koji čekaju na interakciju sa korisnikom, ako i kod koji se izvršava u slučaju *timeout*-a.

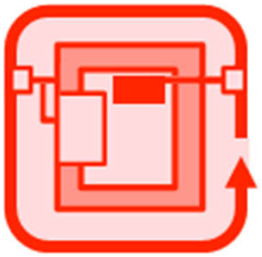




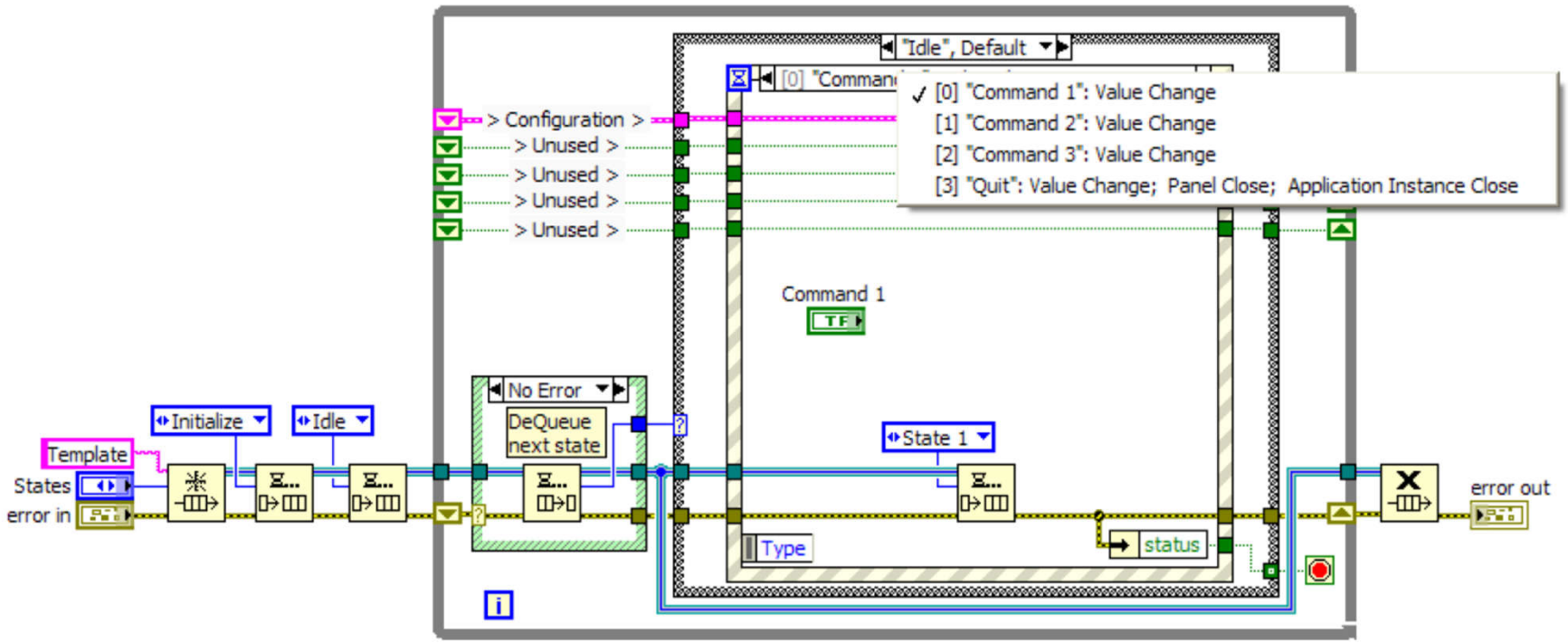




Design Patters – Event Based State Machine



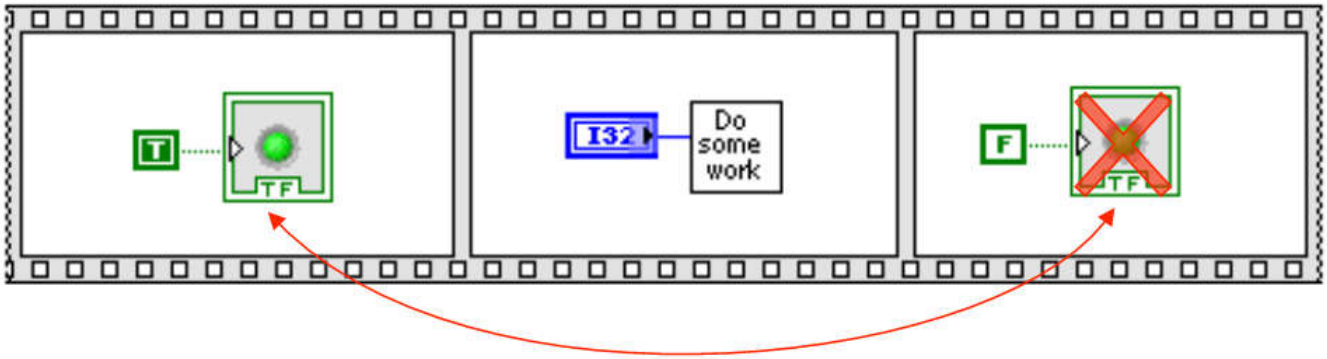
- Poseban slučaj *State Machine*, u kome se proračun sledećeg stanja vrši u *Event Structure*-i.
- Prelaza iz stanja čekanja u neko od stanja sistema određen je pojavom *event*-a.
- Nakon izvršene sekvence stanja, sistem se vraća u stanje čekanja na *event*.



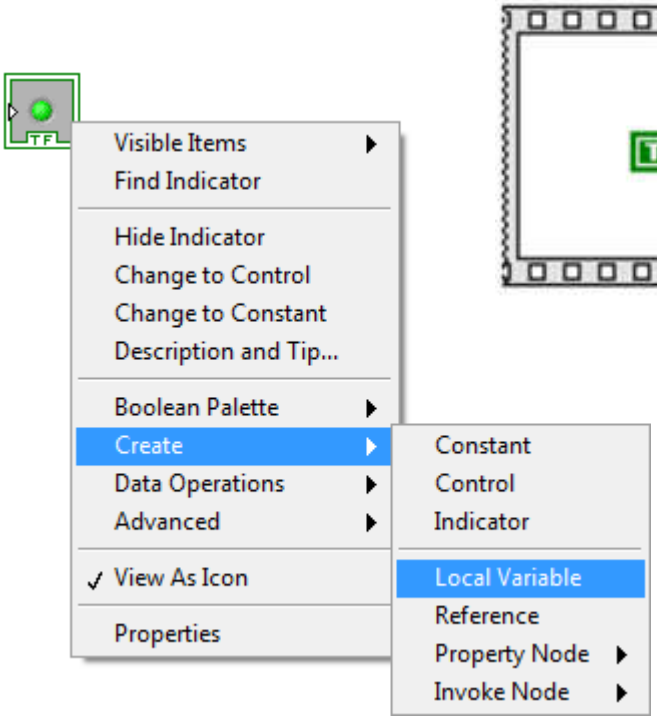
Vežba 17

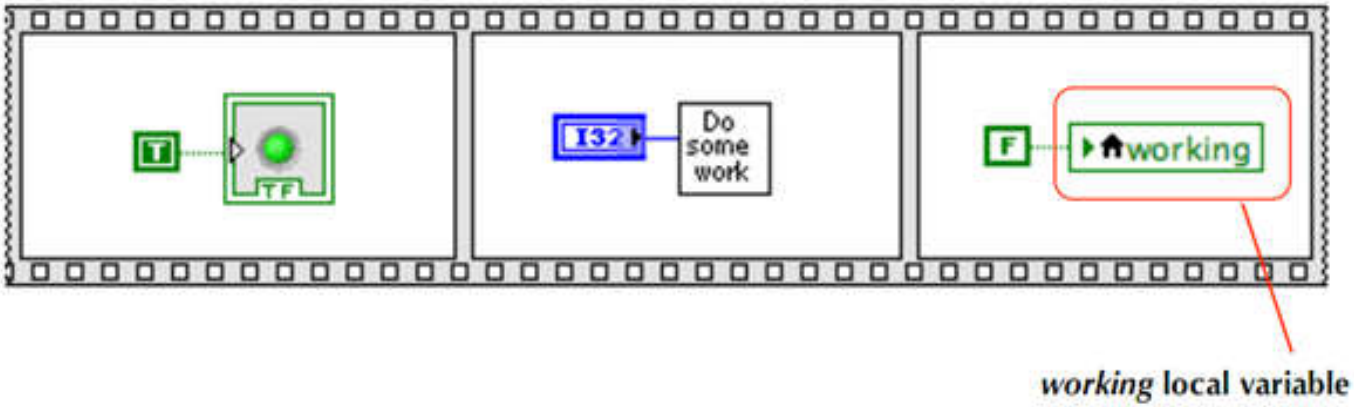
- Modifikovati prethodni program tako da se uključi i *event* struktura sa sledećim *event*-ima:
  - promena parametara simuliranog signala vraća program u stanje inicijalizacije,
  - promena limita postavlja pitanje korisniku da li je siguran da želi da promeni limite, vrši promenu limita ukoliko se korisnik saglasio uz međusobnu zamenu vrednosti limita ako nisu dobro definisani (donji veći od gornjeg),
  - odbija zahteva za zatvaranje prozora,
  - zaustavlja program ukoliko korisnik aktivira taster stop.

Lokalne promenljive



Nije moguće sa istim indikatorom.



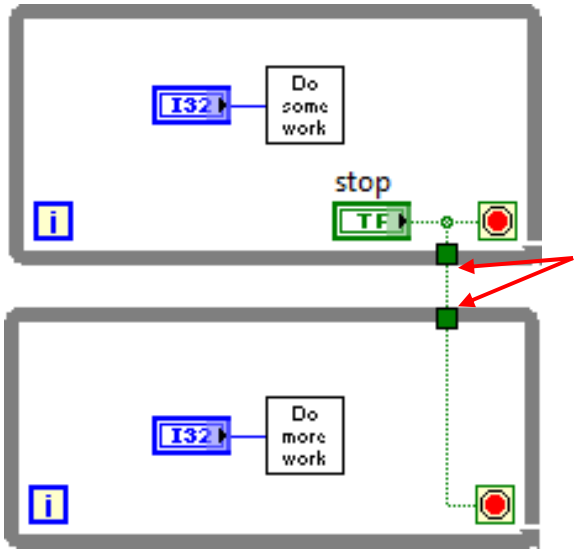


working local variable

- Lokalna promenljiva je povezana sa odgovarajućom kontrolom/indikatorom.
- U lokalnu promenljivu se može upisati vrednosti i iz nje se može iščitati vrednost bez obzira sa tipom terminala sa kojim je povezana

Lokalne promenljive

- Omogućavaju zaustavljanje dve petlje istovremeno.

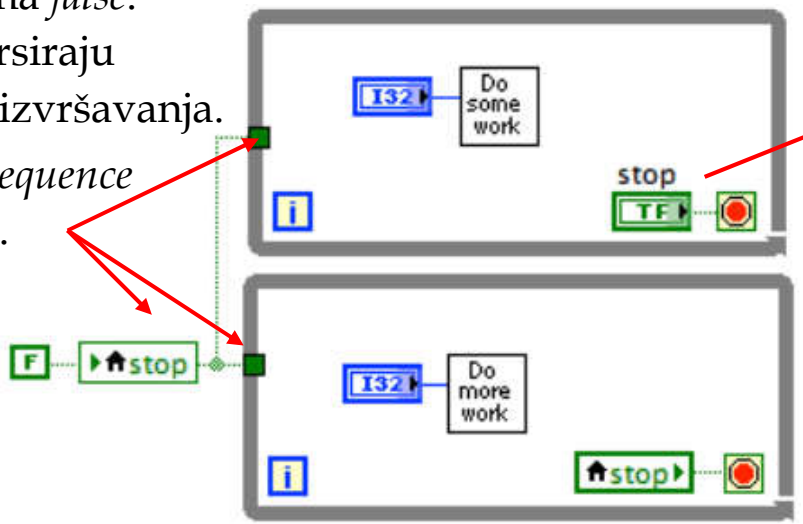


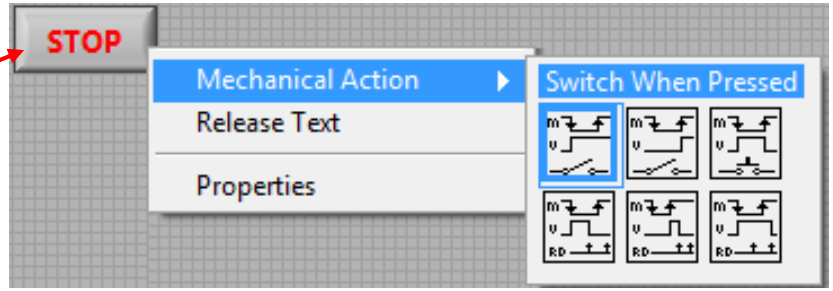
Tunel postaje aktivan tek kada se završi gornja petlja. Tek tada počinje da se izvršava donja petlja i to samo jednom.

Oznaka za lokalnu promenljivu.

Inicijalizacija Stop kontrole na *false*. Tuneli forsiraju redosled izvršavanja.

Umesto *sequence* strukture.

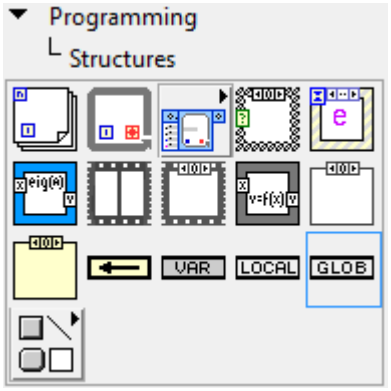




Mora biti *Switch*, jer *Latch* zadržava vrednost dok LabVIEW ne iščita vrednost i resetuje je.

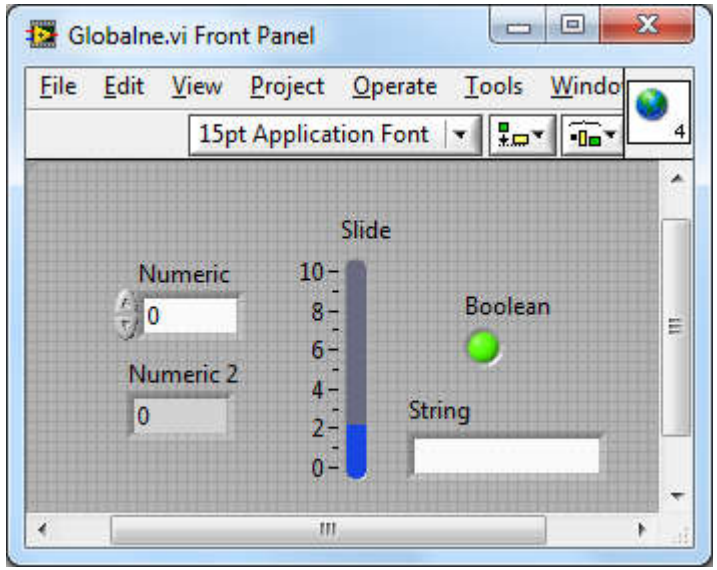
Globalne promenljive

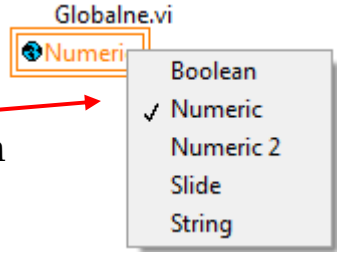
- Lokalna promenljiva je vezana za odgovarajući terminal i važi samo u VI koji sadrži taj terminal (kontrolu/indikator).
- Globalna promenljiva ima opseg važenja na nivou LabVIEW aplikacije. Globalna premenljiva iz jednog projekta se može pozvati u VI drugog projekta.
- Globalne promenljive omogućuvaju razmenu podataka između različitih VI koji se izvršavaju u toku jedne aplikacije.



Fajl Globalne.vi može sadržati više terminala različitog tipa, odnosno predstavlja skup promenljivih.

Nema BP.



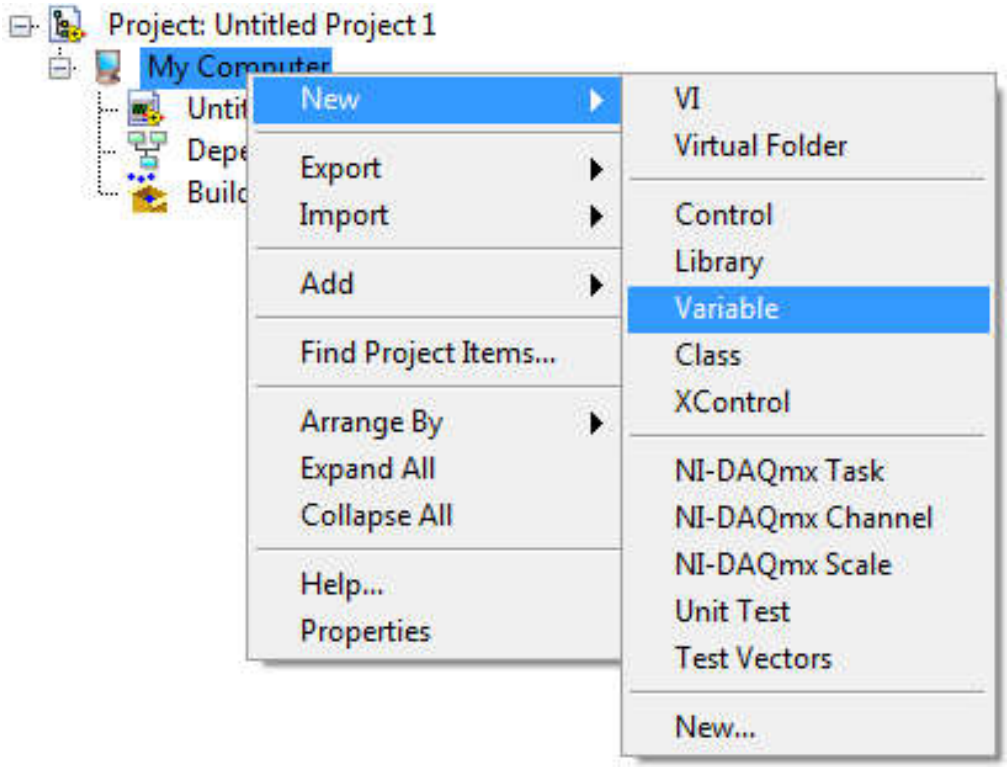


Izbor promenljive iz skupa promenljivih.

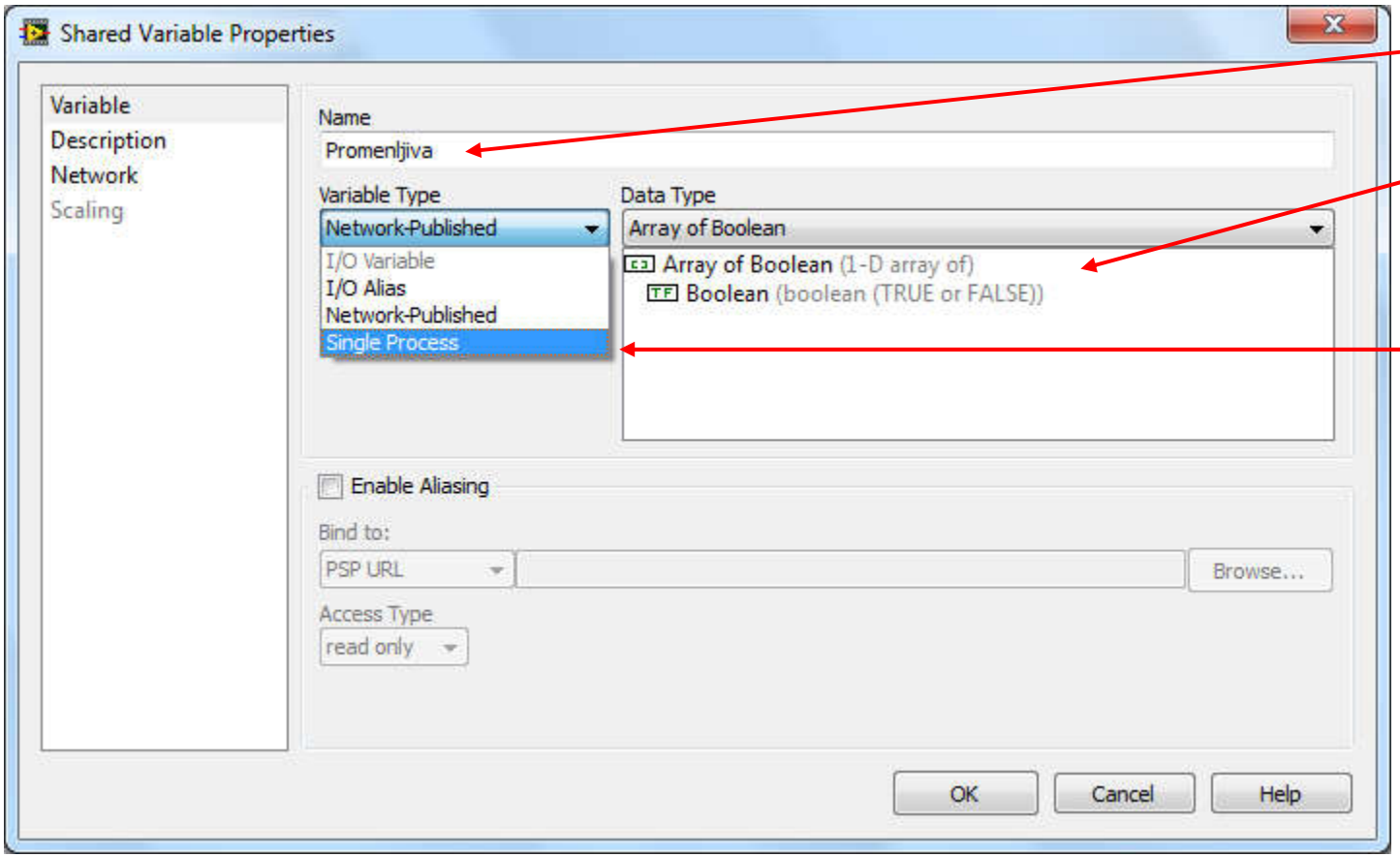
Oznaka za globalnu promenljivu.

Shared variables

- Za razliku od globanlih promenljivih *shared variables* se mogu koristi na više umreženih računara čime se omogućava mrežno deljenje resursa.
- Vezana je za projekat.



Shared variables

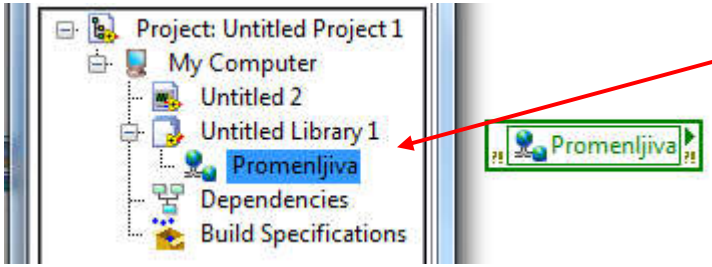


Naziv promenljive

Tip podatka promenljive

Single process – isto što i globalna promenljiva, jedino što vrlo lako postaje vidljiva na mreži, bez izmena koda.

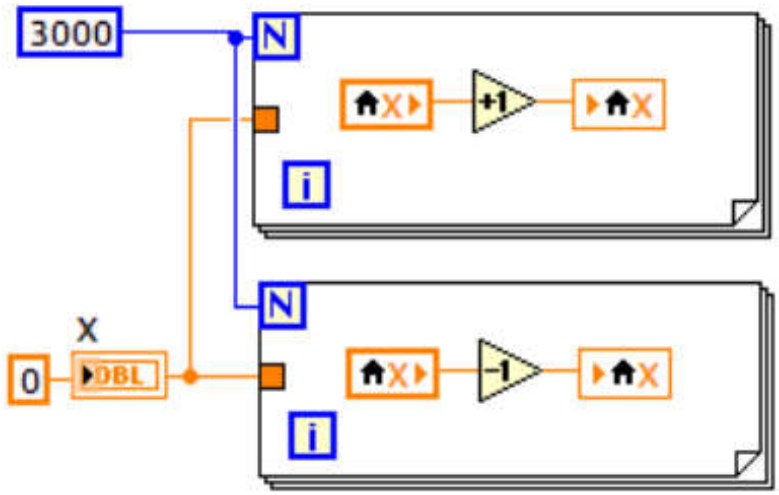
Network-Published – vidljiva na mreži.



Prevlačenje na *BP*

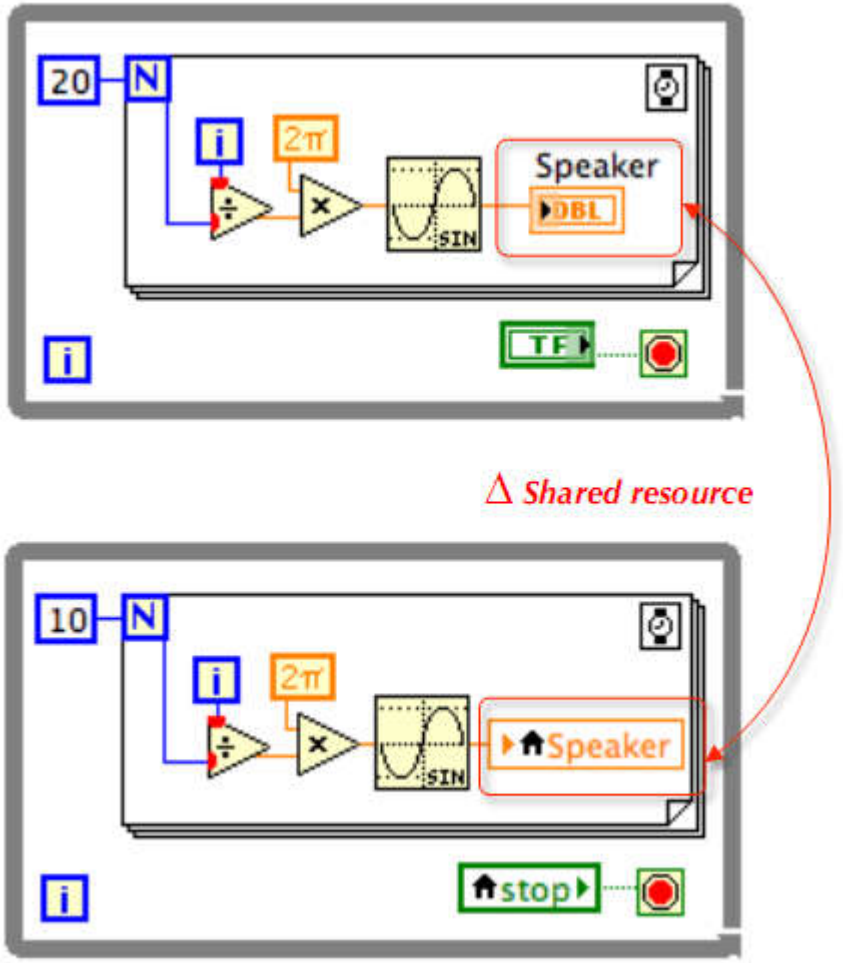
Race condition

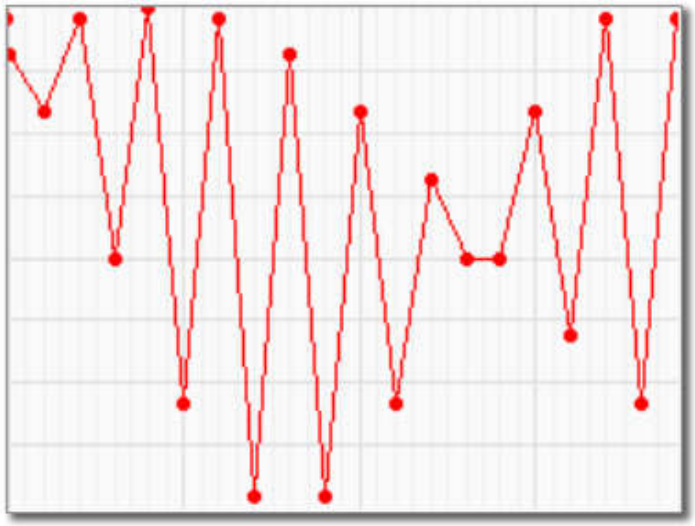
- Paralelno programiranje omogućava brže izvrašavanje nekog zadatka, međutim mogu se javiti neželjeni efekti kao što je istovremeno pristupanje zajedničkom resursu.



Nemoguće je utvriti vrednost promenljive *x* nakon izvršavanja BP-a. U 5 uzastopnih izvršavanja dobijaju se slučajne vrednosti: 107, -848, -192, 598, 415.

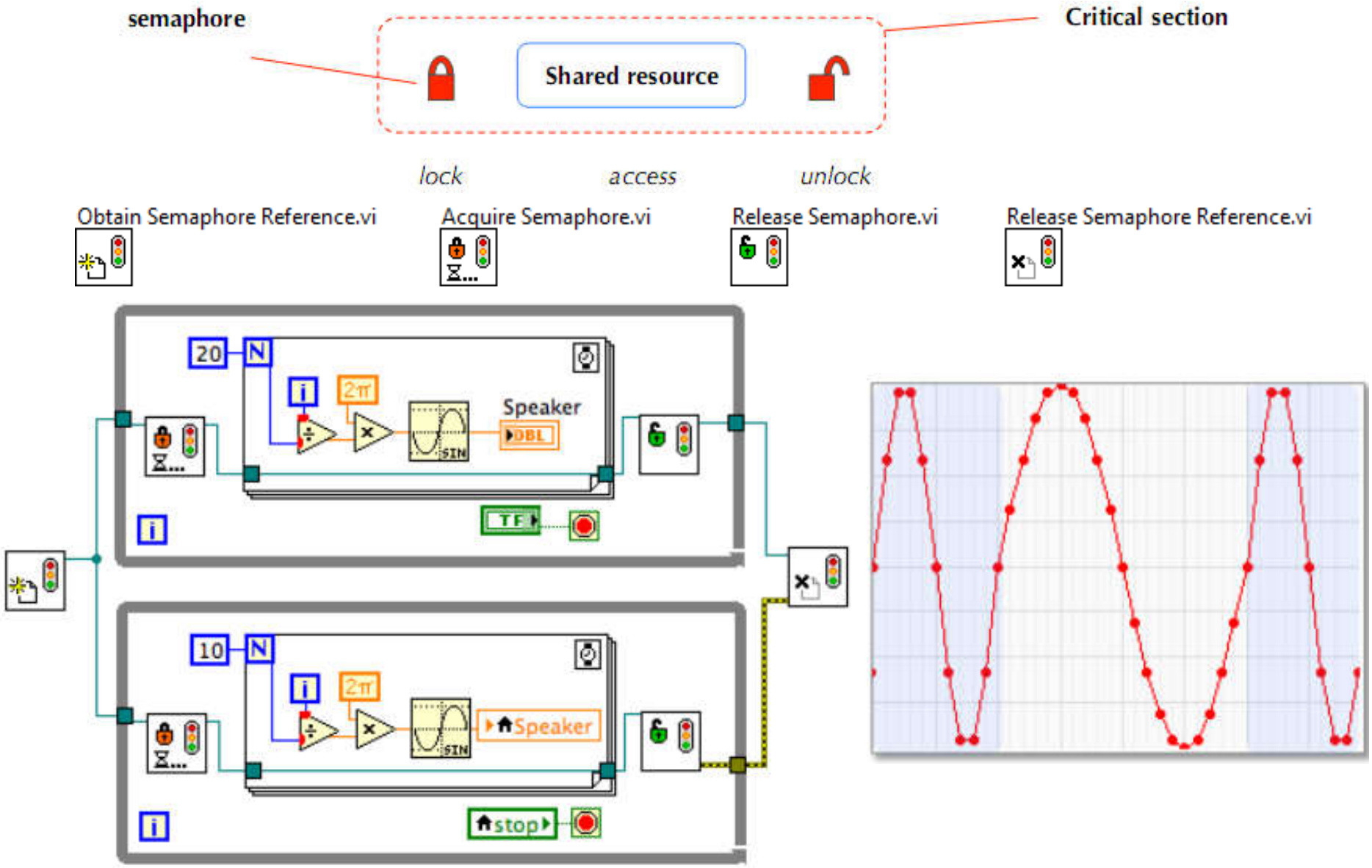
Race condition





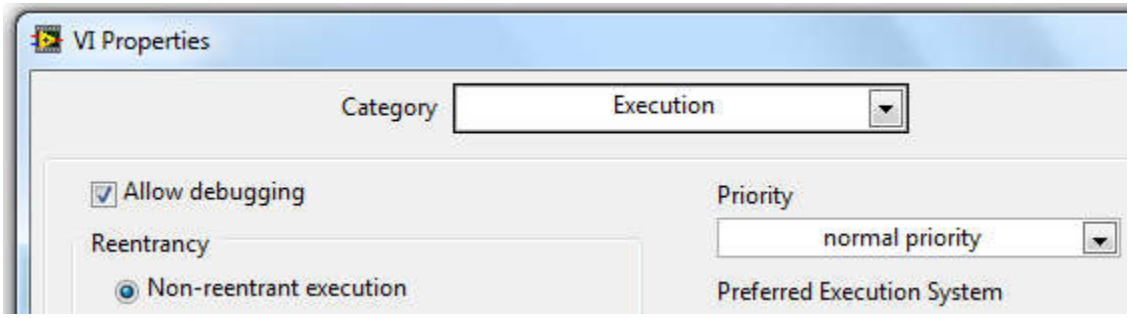
Race condition – rešenje 1

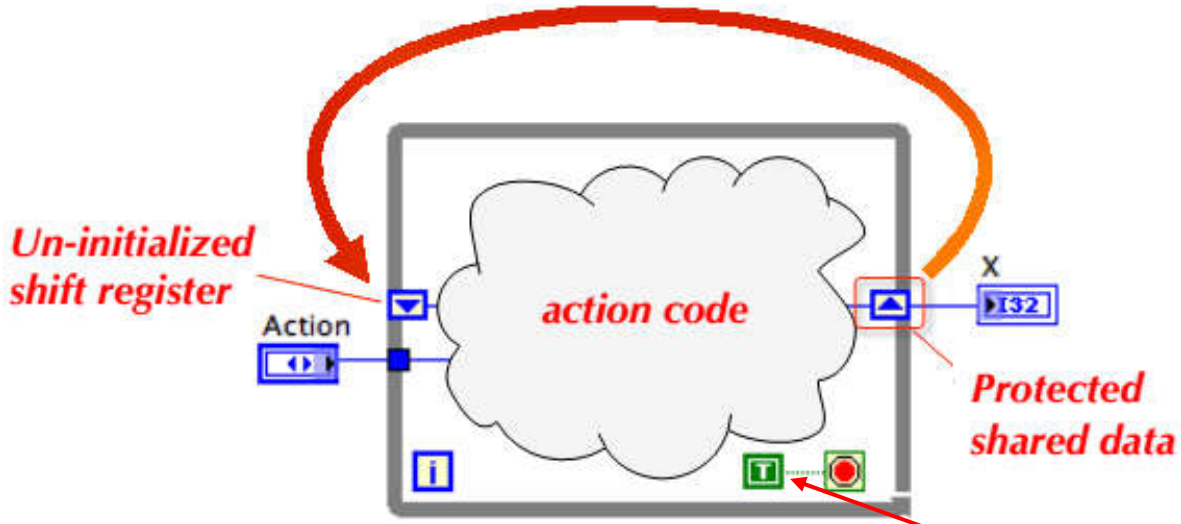
- Prvo moguće rešenje su semafori.



Funkcionalne globalne promenljive

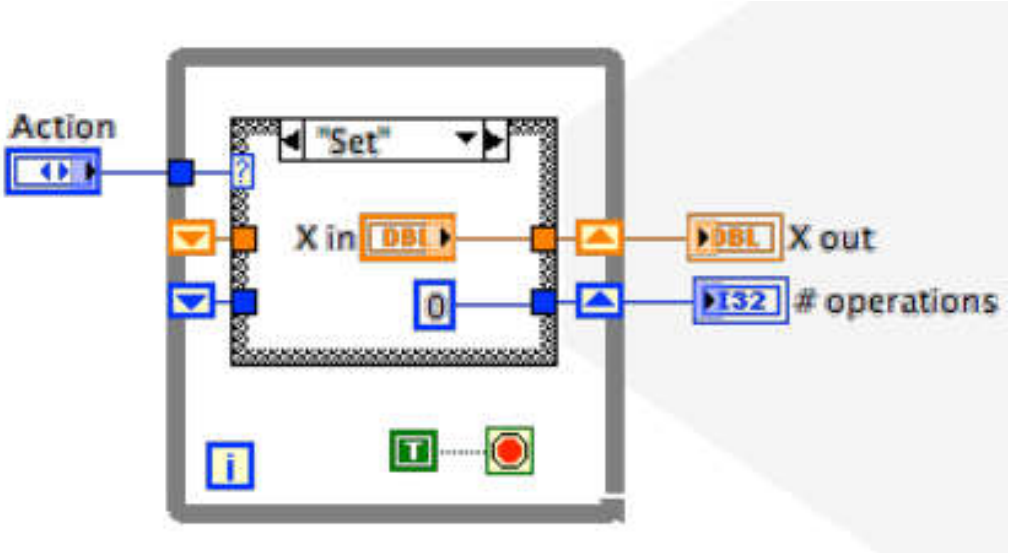
- Drugo moguće rešenje – Funkcionalne globalne promenljive.
- *FGV - Functional global variable*.
- Koristi se neinicialiozovani *Shift Register*.
- Korisnik definiše akciju koja će se izvršiti nad zaštićenim podatkom.
- VI mora biti *non-reentrant*.

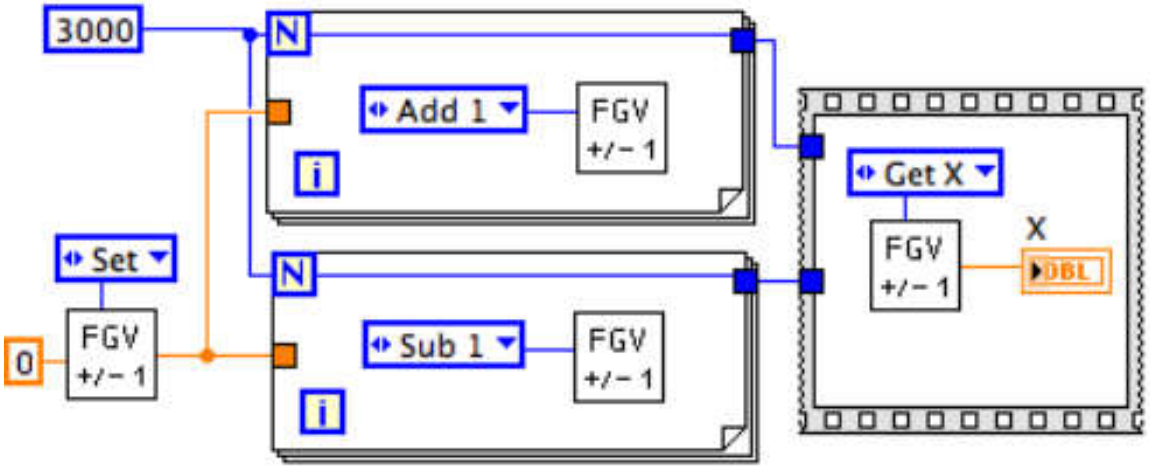




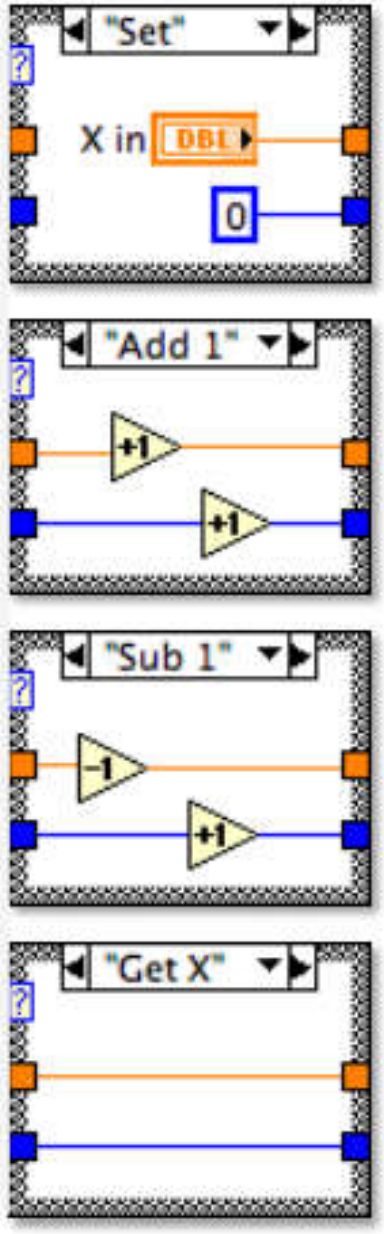
Izvršava se samo jednom

Funkcionalne globalne promenljive



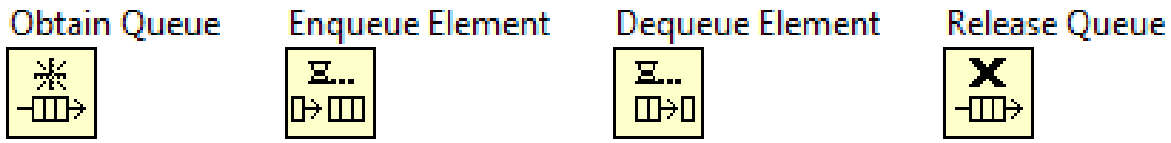


X = 0



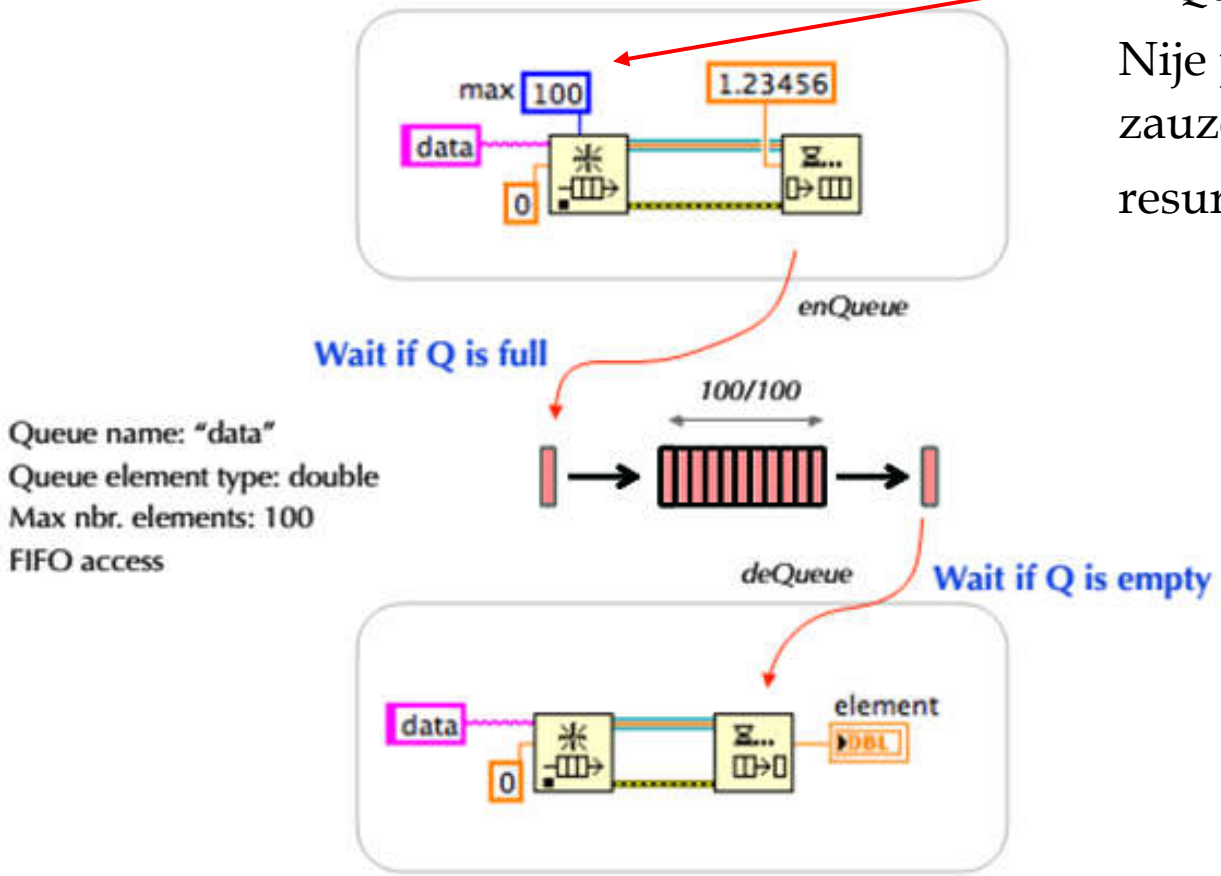
Queue

- Omogućavaju razmenu podata između dve petlje.
- Jedan *Master* i jedan *Slave*. Za više *Slave*-a koriste se *Notifier*-i.

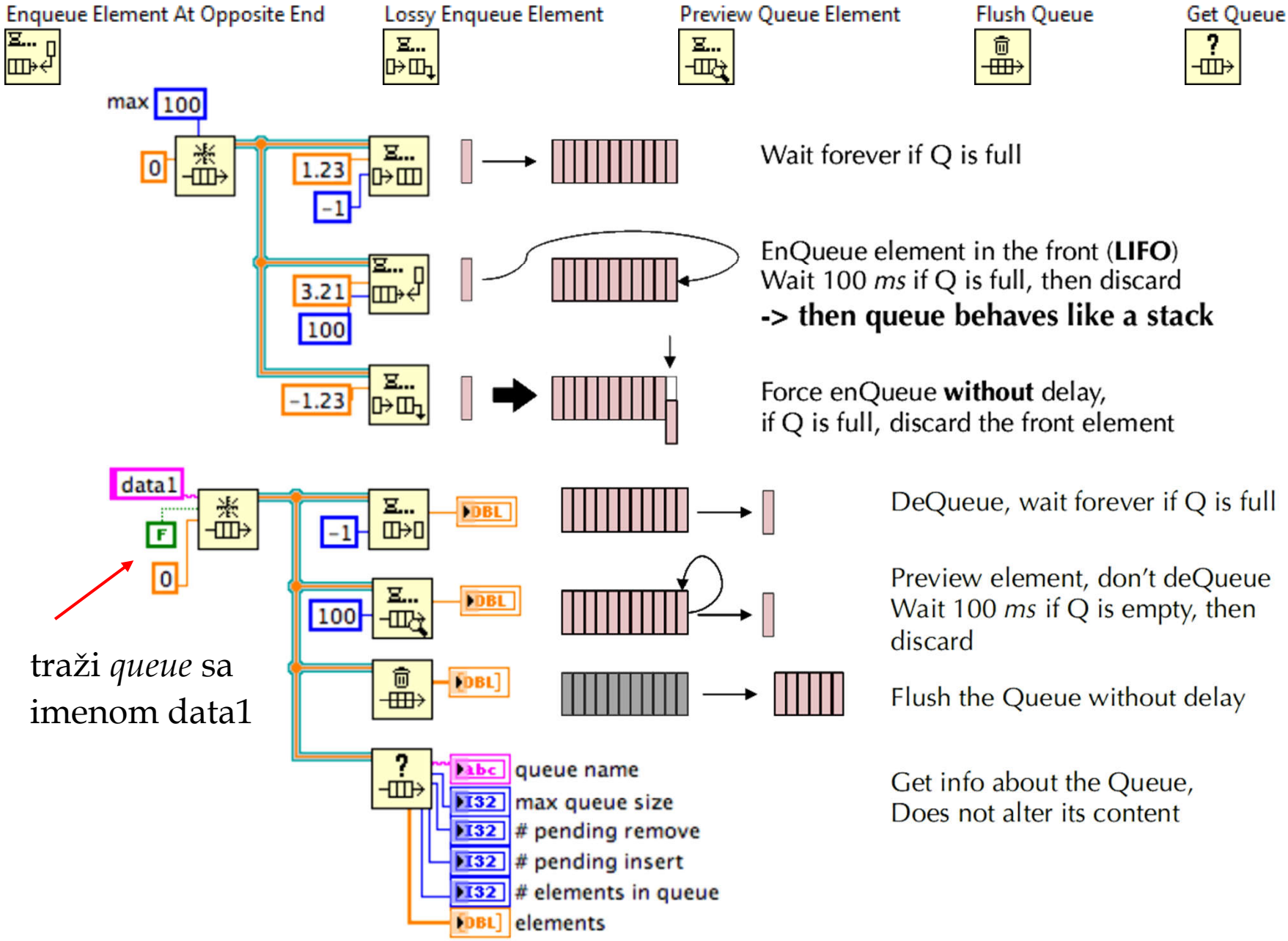


-1 Queue je neograničen.

Nije preporučljivo, može zauzeti sve memorijske resurse.

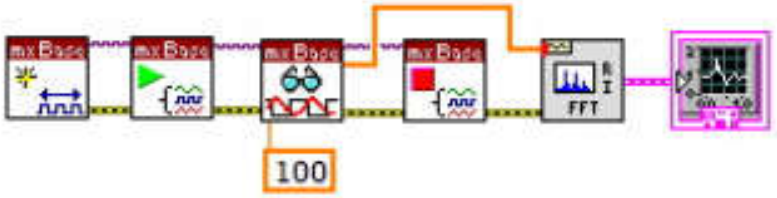


Queue – dodatne funkcije



Design Patterns - dodatno

- *Design Patterns* ili programski šabloni su standardni način pisanja koda u LabVIEW koji se preporučuju:
  - *One shot,*
  - *One loop,*
  - *State machine,*
  - *Event loop* (*Event programing*),
  - *Multiple loops,*
  - *Producer/consumer.*

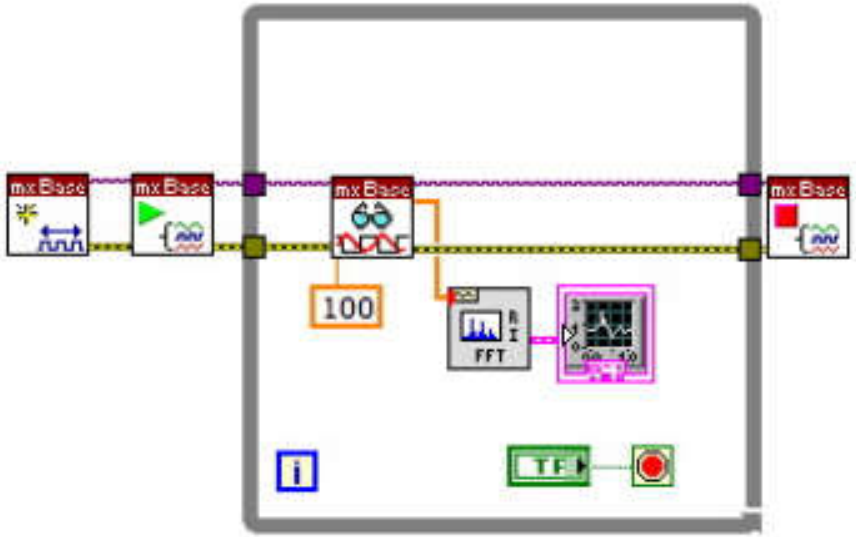


One shot ili još i *Simple Design Pattern*

One shot

```
Config();
Start();
Acquire(100);
Stop();
PostProcess(FFT);
Display();
```

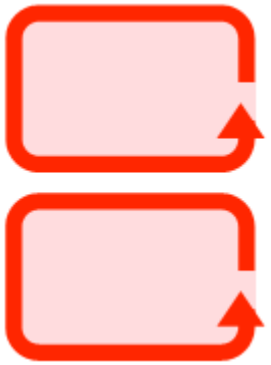
One loop ili još i *General VI Framefork*

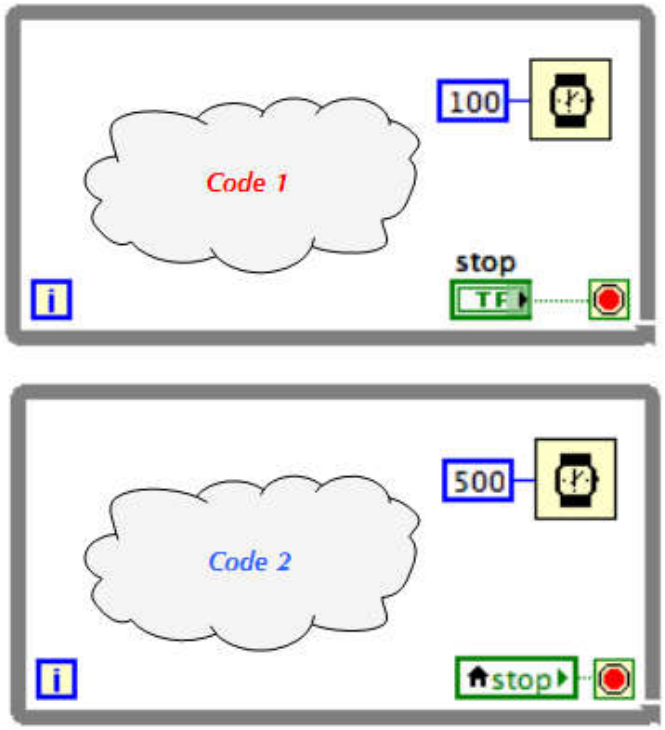



```
Config();
Start();
While (!Stop()) {
   Acquire(100);
   PostProcess(FFT);
   Display();
}
Stop();
```

Design Patterns – Multiple loops

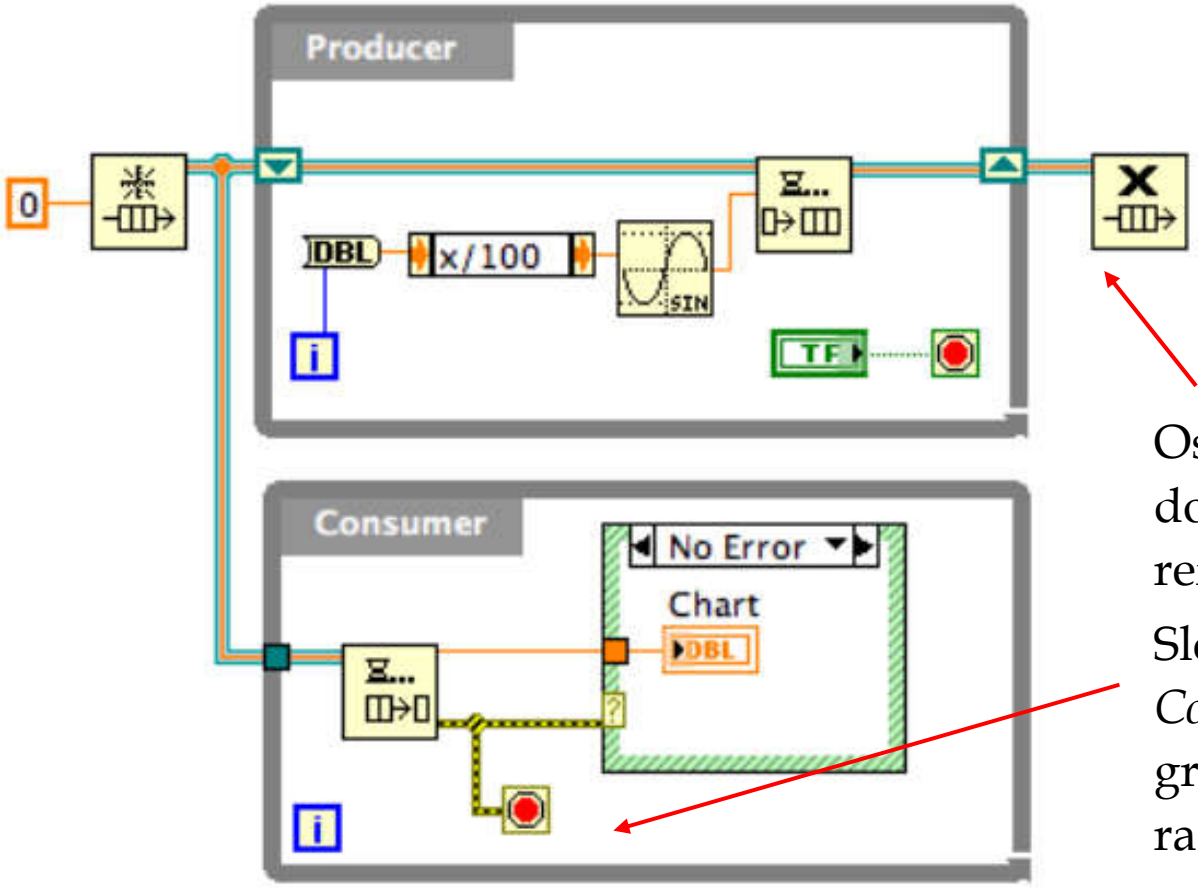
- Nezavisan, paralelan rad više petlji, koje ne moraju da dele resurse.
- Ukoliko petlje dele resurse, potrebno je koristi neki od mehanizama sprečavanja *Race Condition.*
- Petlje nisu sinhronizovane (postoji moućnost sinhornizacije koućenjem *subpalette Programming » Structures » Timed Sturctures*).
- LabVIEW, ukoliko postoje mogućnosti, svaku petlju dodeljuje drugom procesoru.





Design Patterns – Producer / Consumer

- *Master* (*Producer*) generiše podatke i to može biti i asinhrono.
- *Slave* (*Consumer*) čeka da podaci budu dostupni, a zatim ih koristi.
- Razmena podataka između dve petlje se vrši pomoću *queue*.

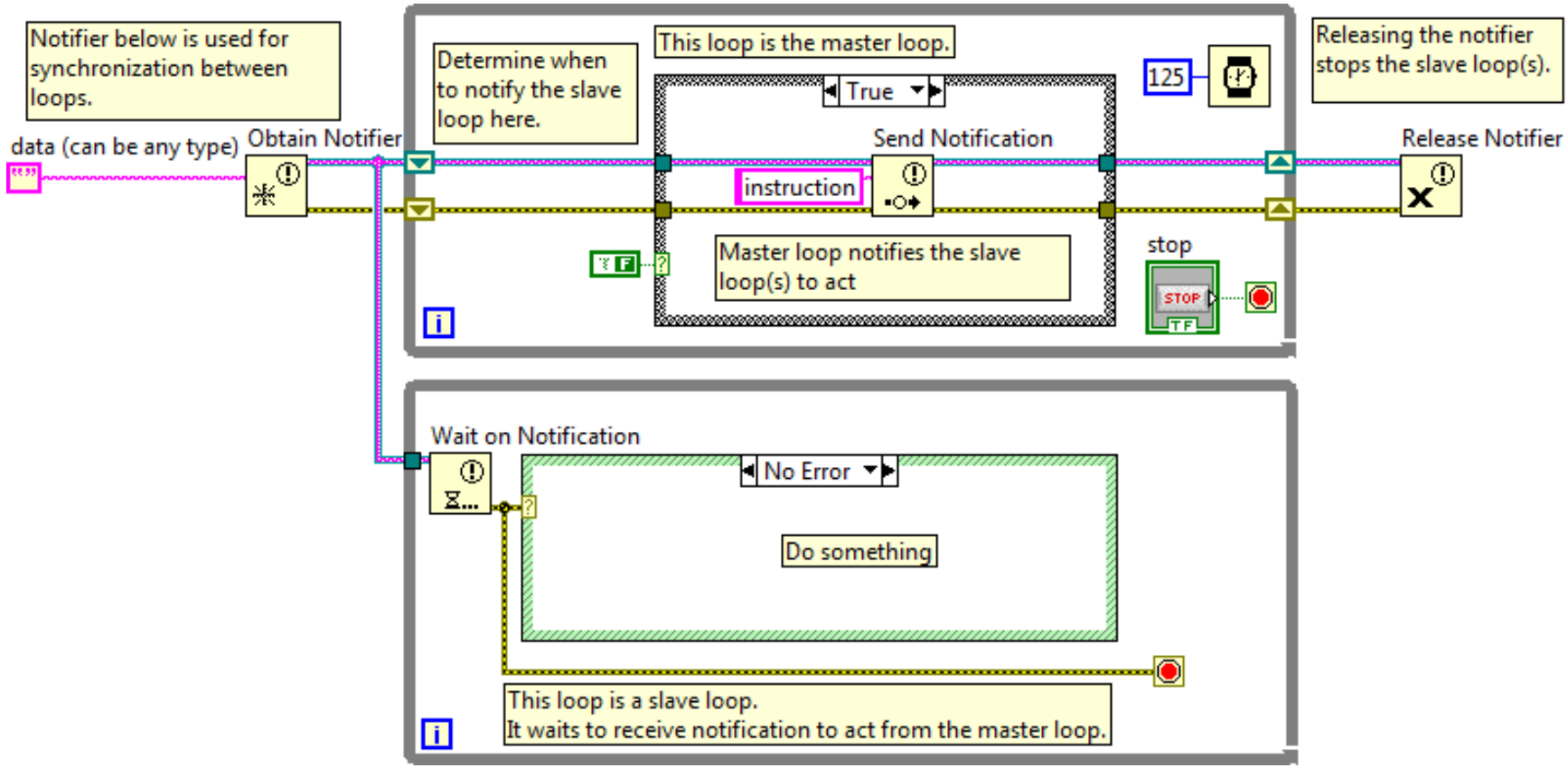


Oslobađa se memorija dodeljena *queue*, a time i referenca na *queue*.

Sledeći poziv *Dequeue* u *Consumer* petlji generiše grešku i ona prestaje sa radom.

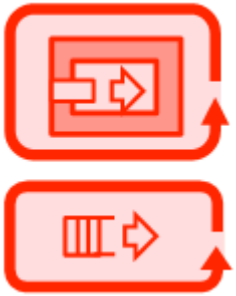
Design Patterns – Producer / Consumer with Notifiers

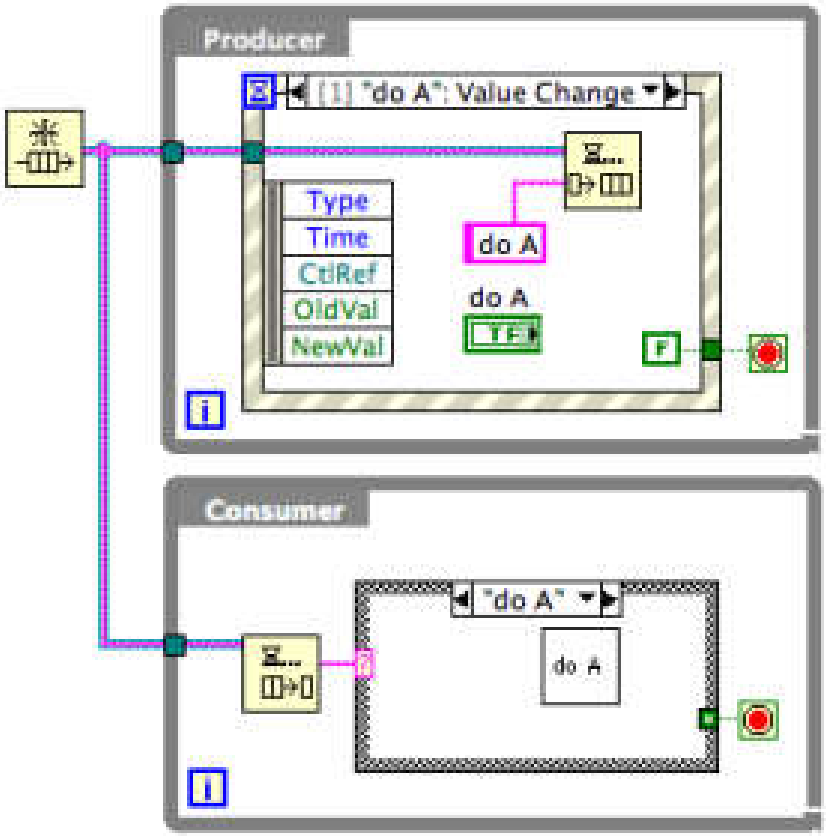
- *Notifiers* – deo BD-a šalje poruku (koja može biti bilo koji tim podataka) drugom delu BD-a ili SubVI. Kada odredište primi poruku (Notification) nastavlja sa izvršavanjem programa.
- Može sprečiti *Race Condition*. Za razliku od *Queue, Notifiers* omogućava komunikaciju sa više petlji istovremeno (korisnik podataka iz *Queue*, briše podatak kada ga pročita).
- *Master/Slave* programski šablom se može realizovati pomoću *Notifiers*.



Design Patterns – Producer / Consumer Event Loop

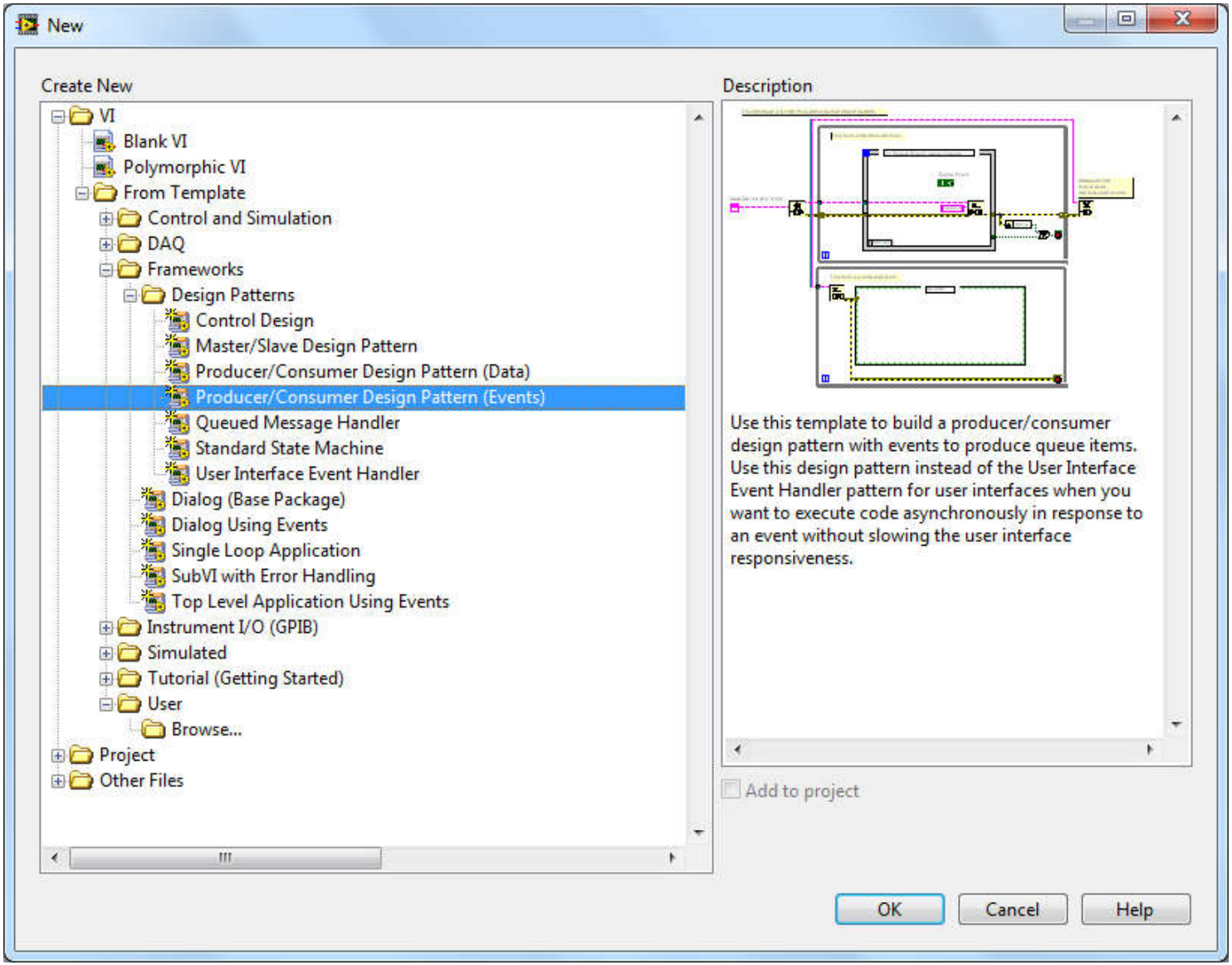
- *Producer* registruje *event* , ali ga ne obrađuje, već šalje podatke o *event*-u *Consumer*-u pomoću *queue*-a.
- Registracija *event*-a vrlo kratko traje, sva obrada je odvija u *Consumer* petlji.





Design Patterns

- *New VI from template.*



Vežba 18

- Za prethodni zadatak obezbediti da se analiza signala obavlja u posebnoj petlji, korišćenjem reda (*queue*).

Akvizicija signala

- LabVIEW omogućava akviziciju sa različitih uređaja.

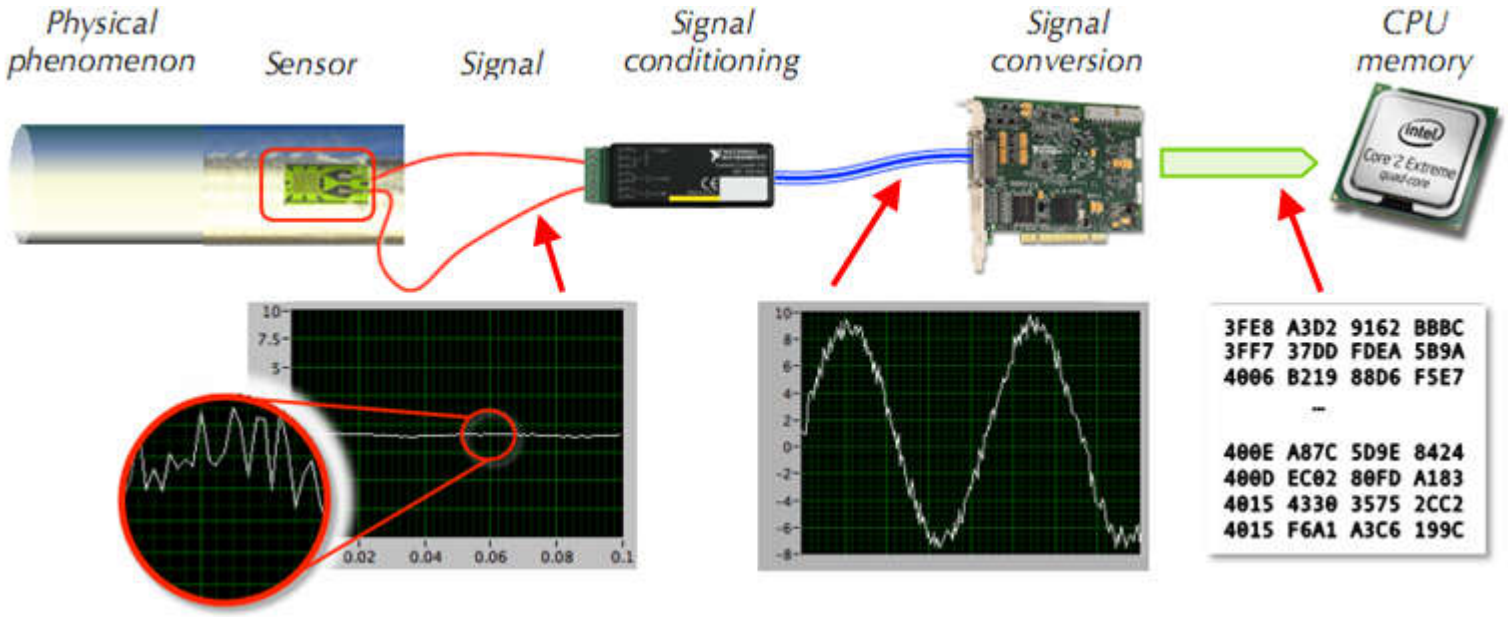
GPIB instrument

DAQ card

USB device

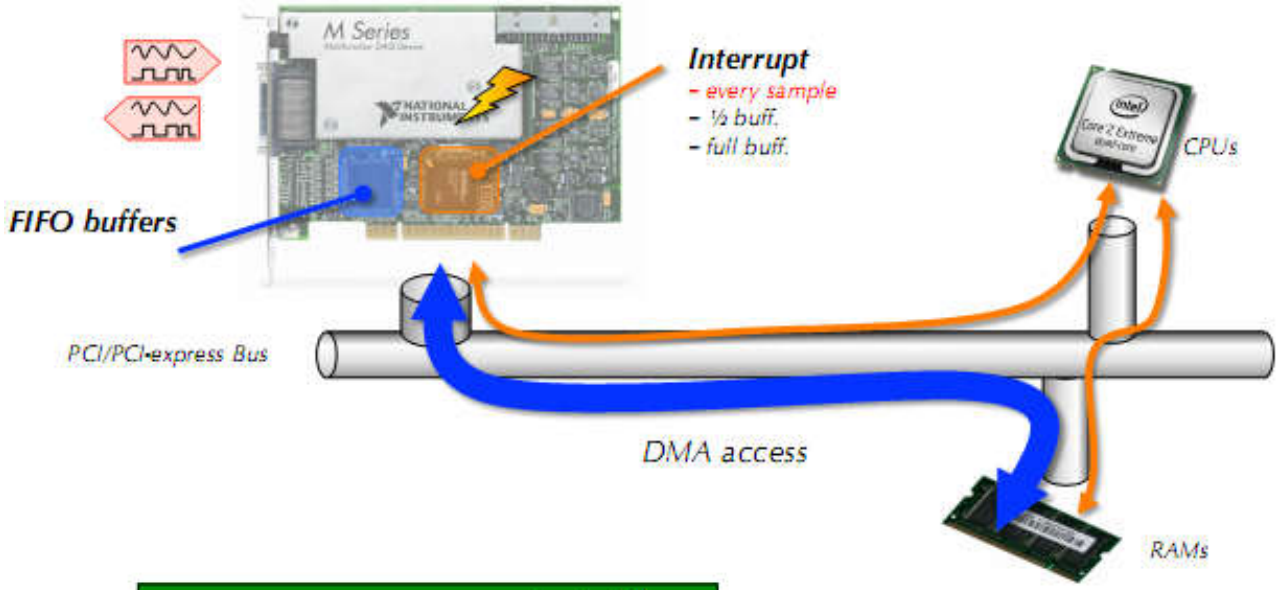
cRio controller

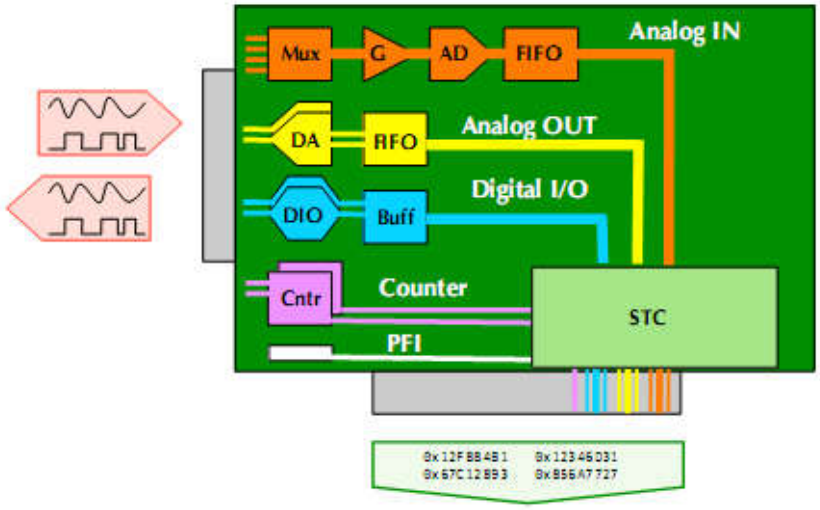
- Od fizičke pojave do računara.



Akvizicija signala

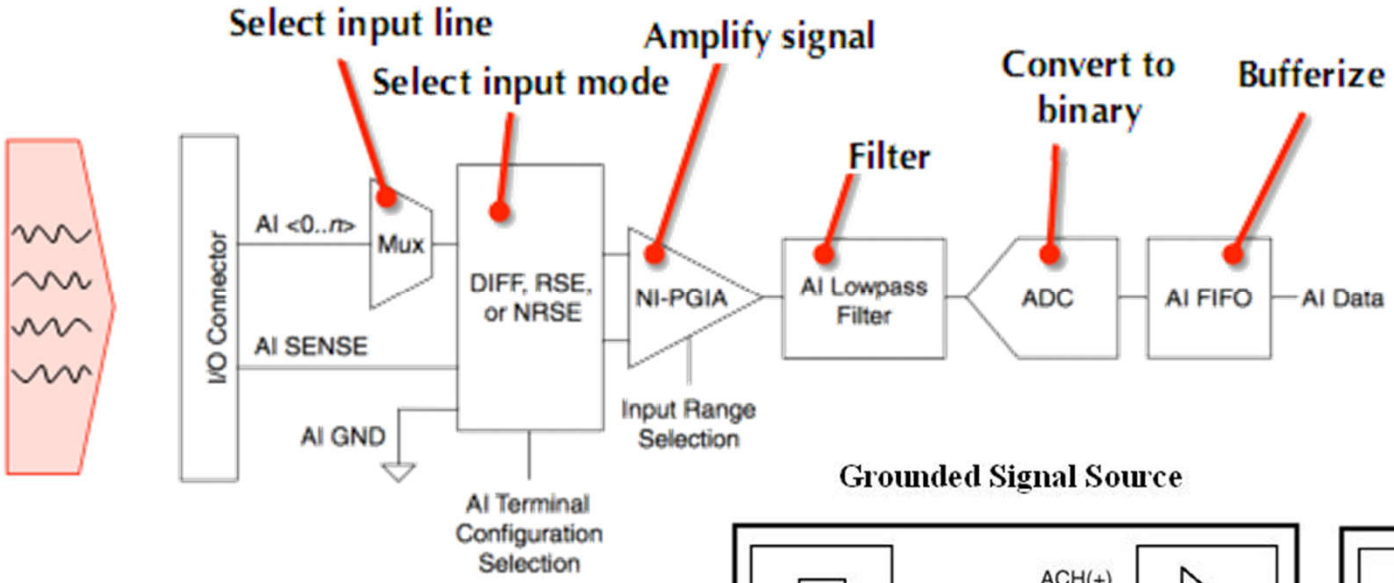
- Prenos digitalnog računara do CPU-a.

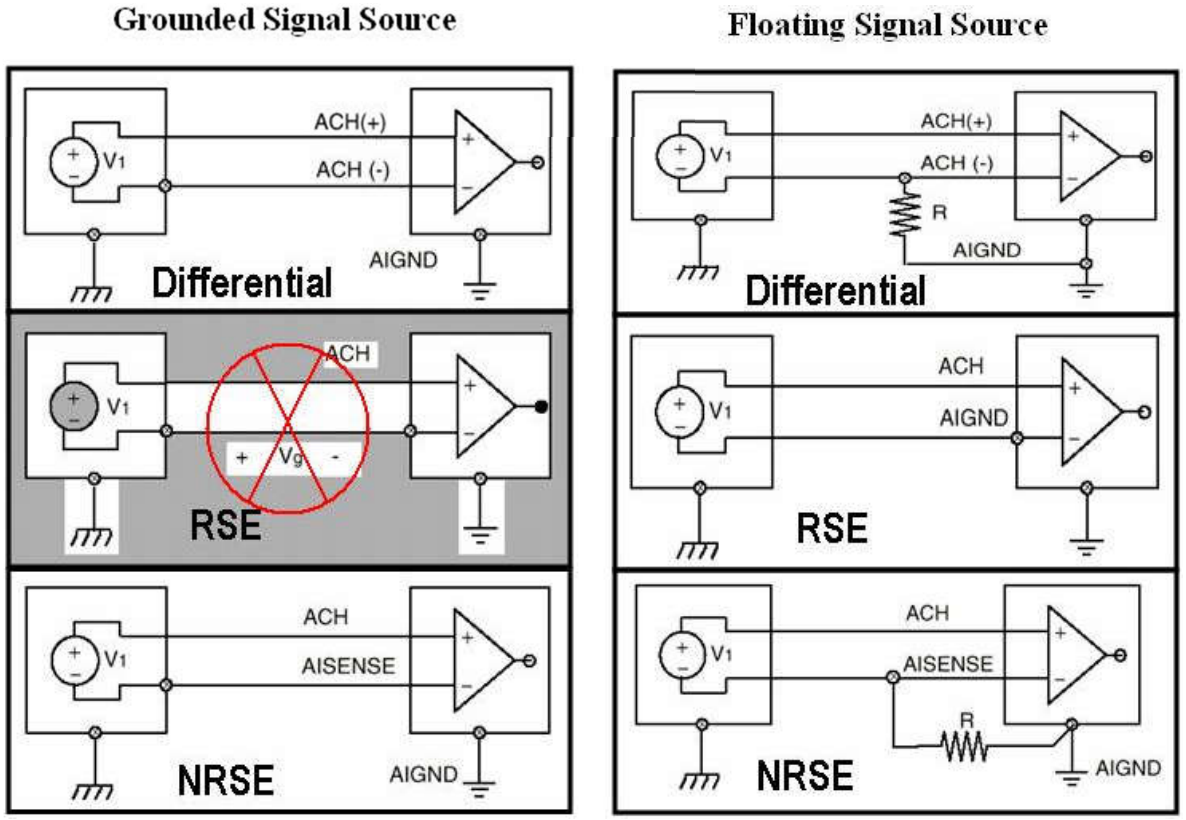




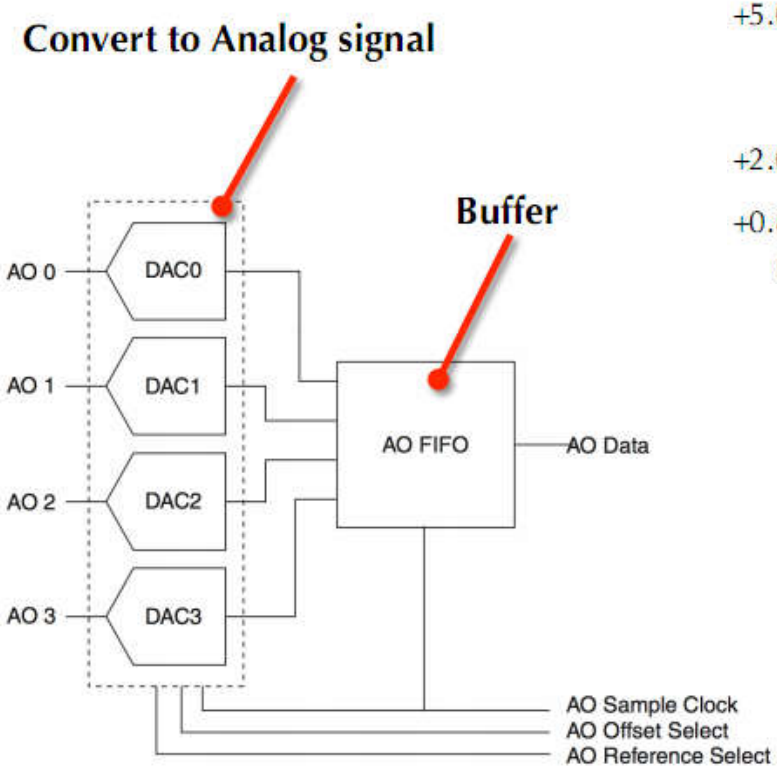
- Veliki broj NI DAQ (*Data Acquisition*) kartica je multifunkcionalan.

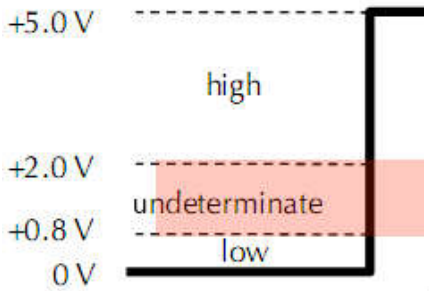
Analogni ulazni signali

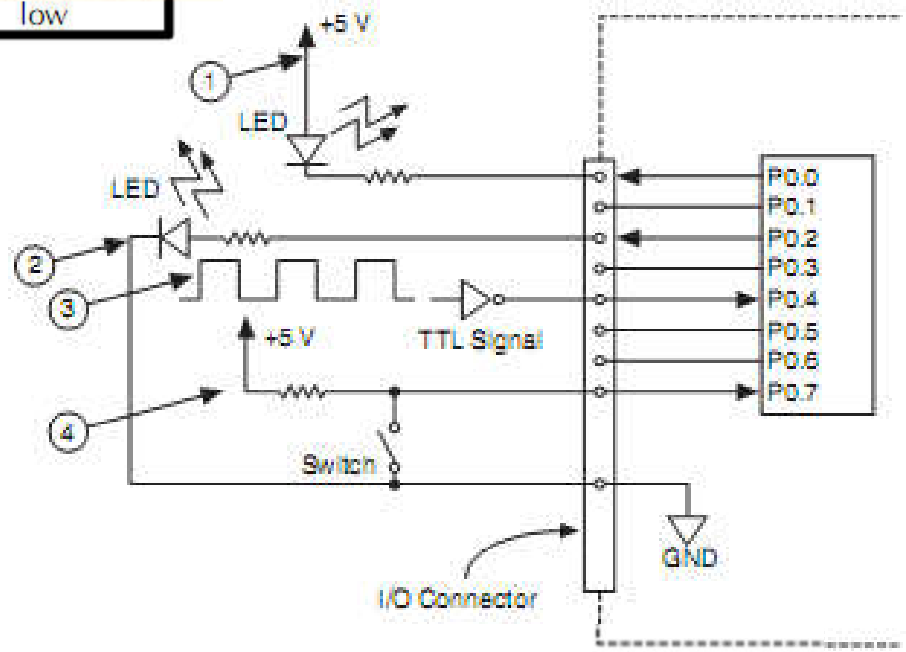


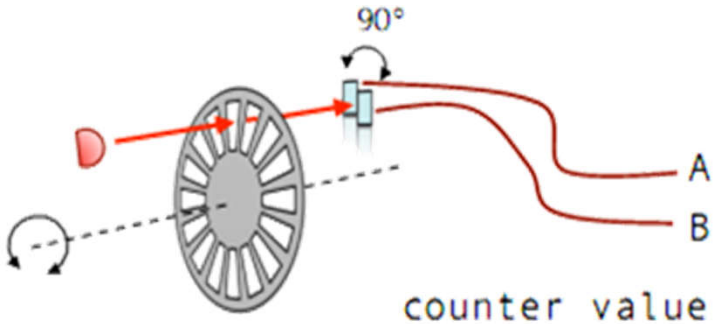


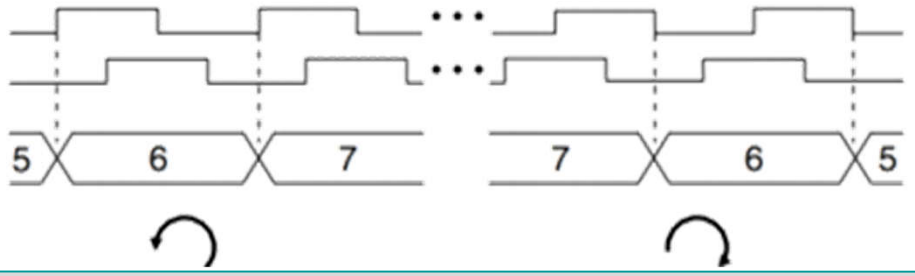
Analogni izlazi, digitalni signali, brojači









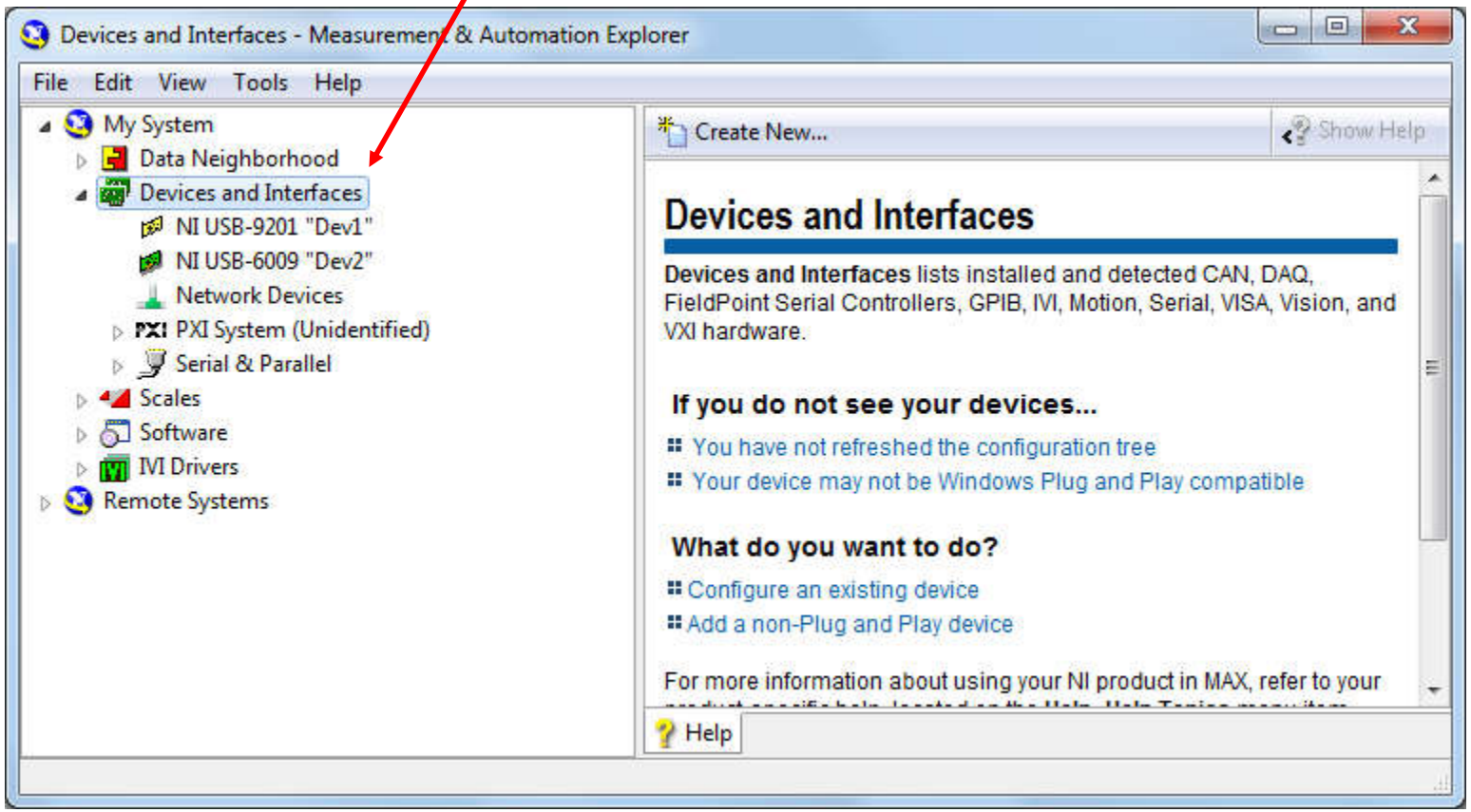


MAX – Measurement and Automation Explorer

- Samostalna aplikacija koja omogućava pregled i podešavanje dostupne merno-upravljačke opreme, pre svega opreme čije je proizvođač *National Instruments*.

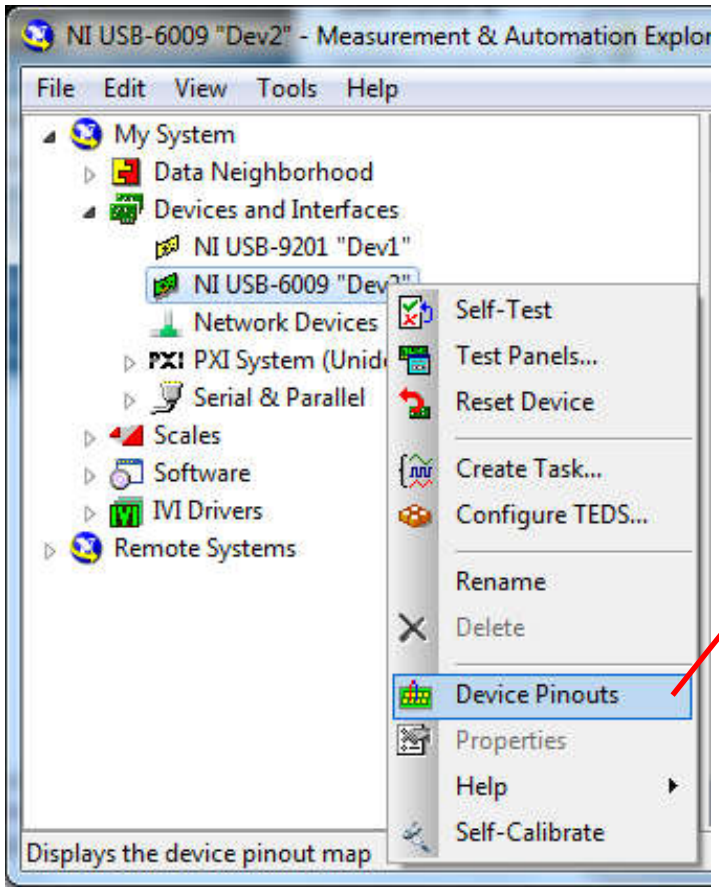
Device and Interfaces – izlistani su svi dostupni uređaji.

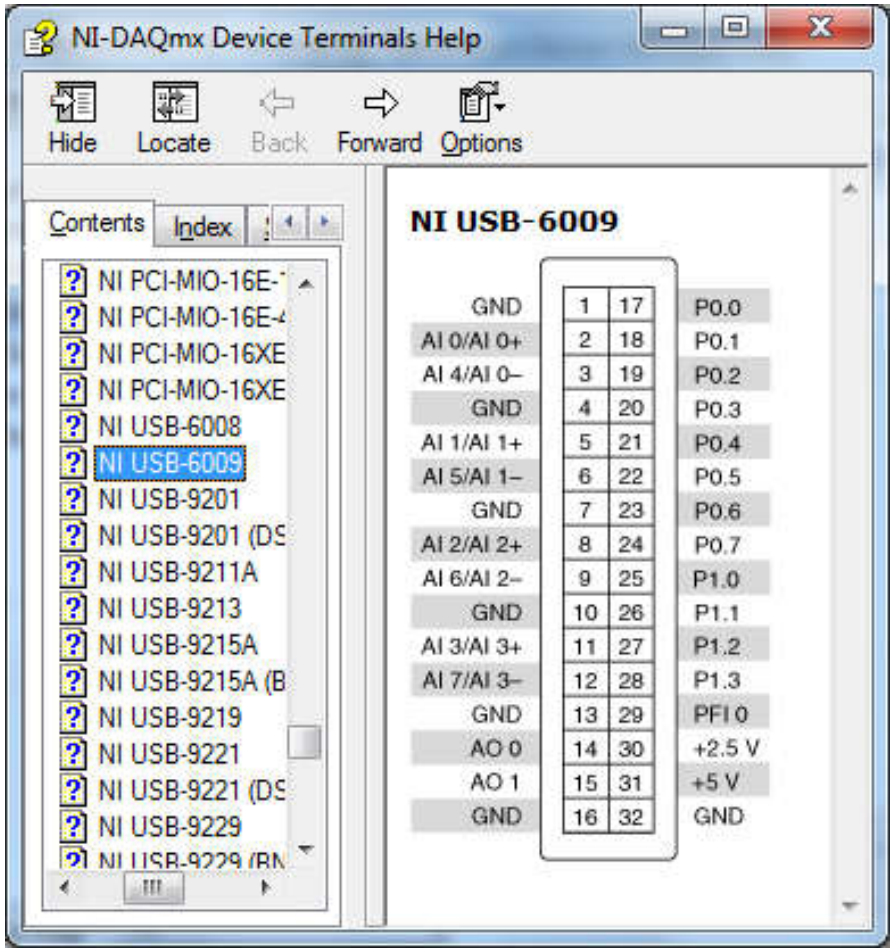
Remote System – svi mrežni uređaji.



MAX – Measurement and Automation Explorer

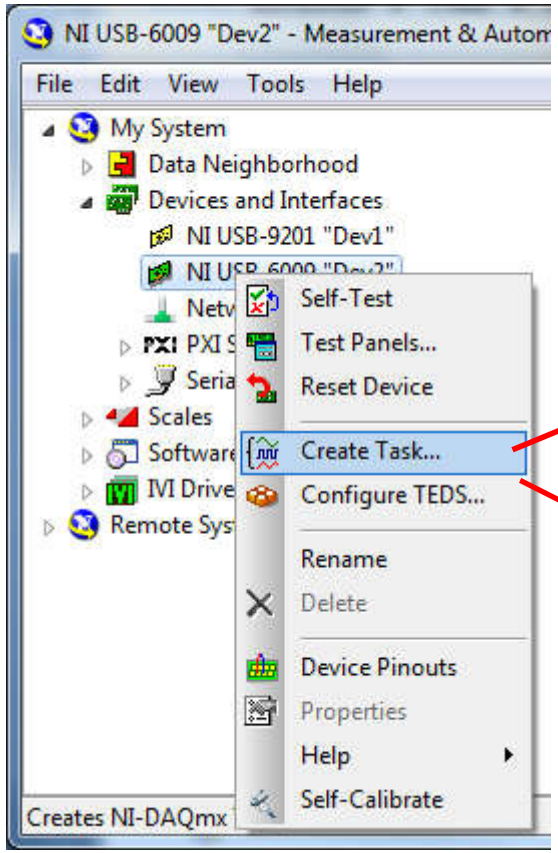
- Pregleda rasporeda konektora uređaja.

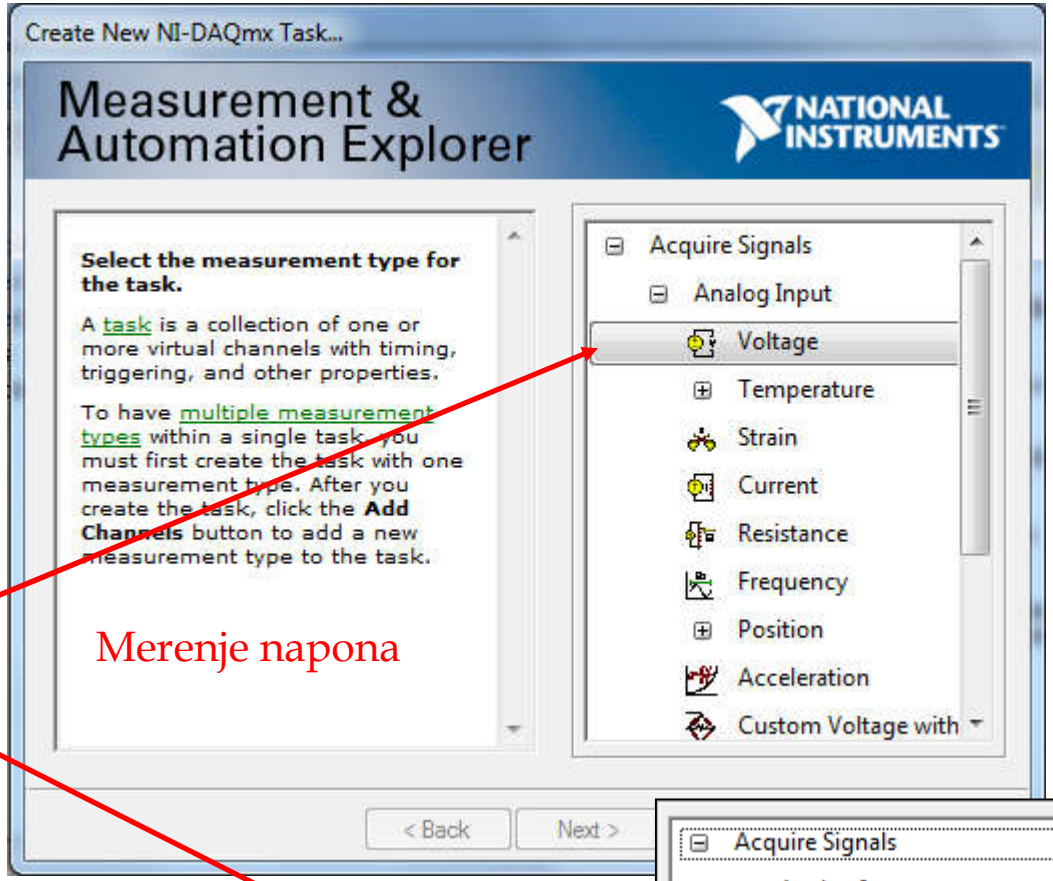


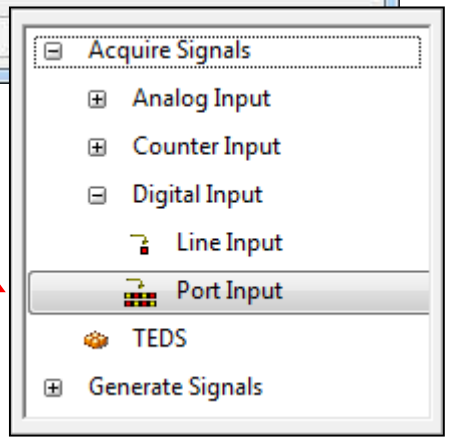


MAX – Measurement and Automation Explorer

- Kreiranje *Task*-a.



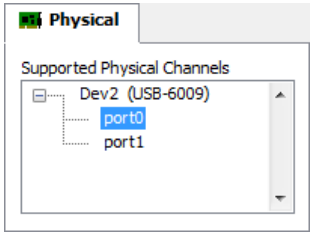


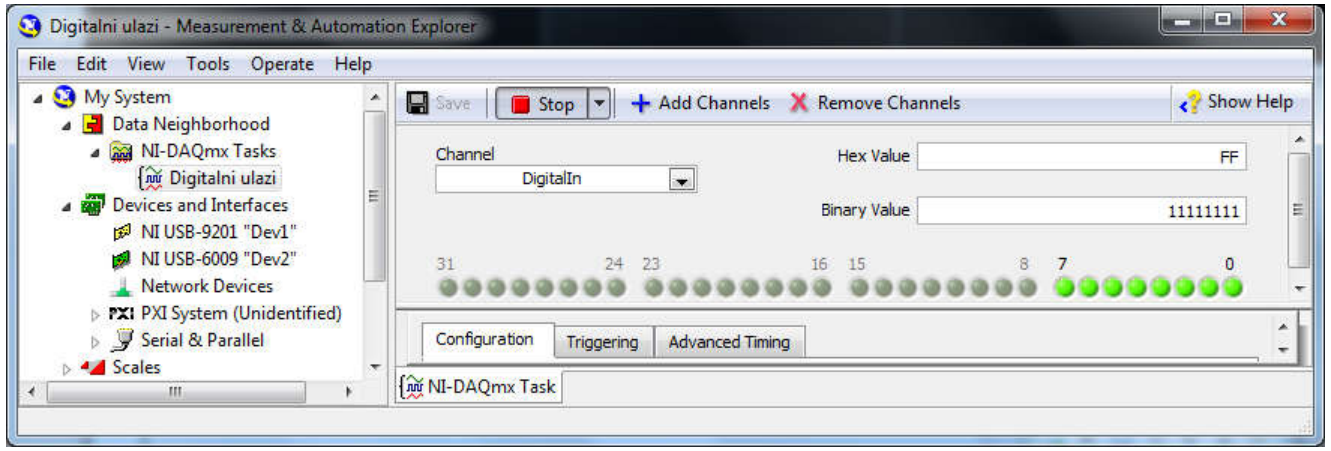


Digitalni ulazi

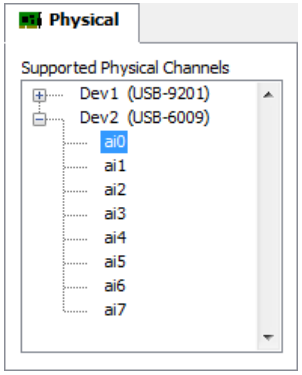
MAX – Measurement and Automation Explorer

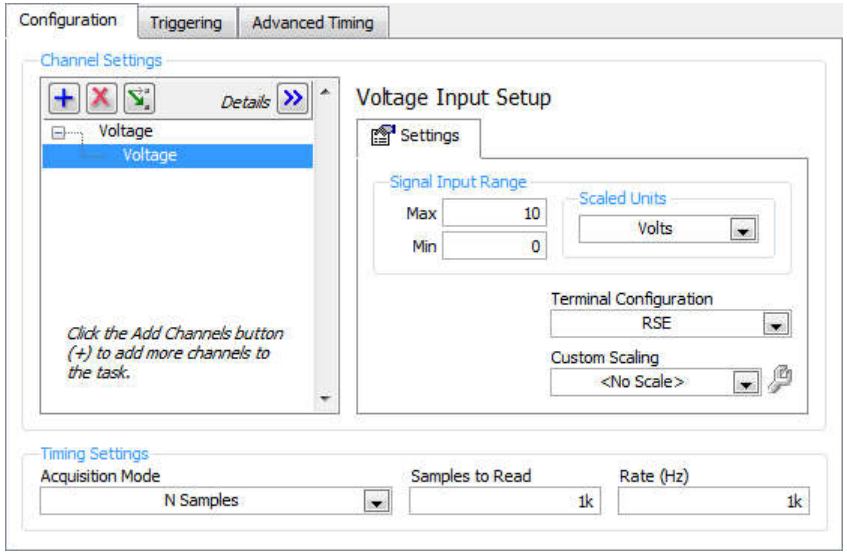
- Digitalni ulazi

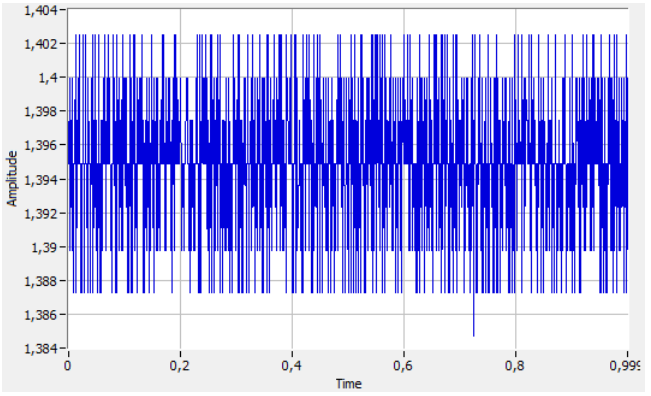




- Analogni ulazi



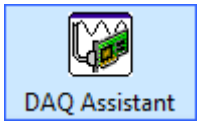


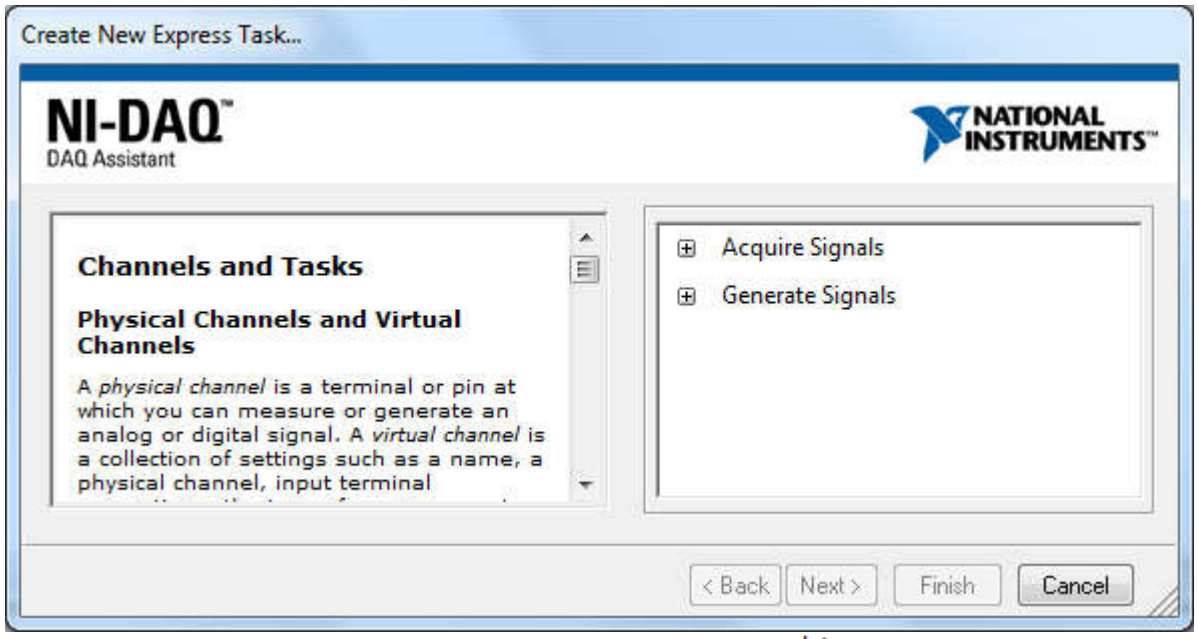


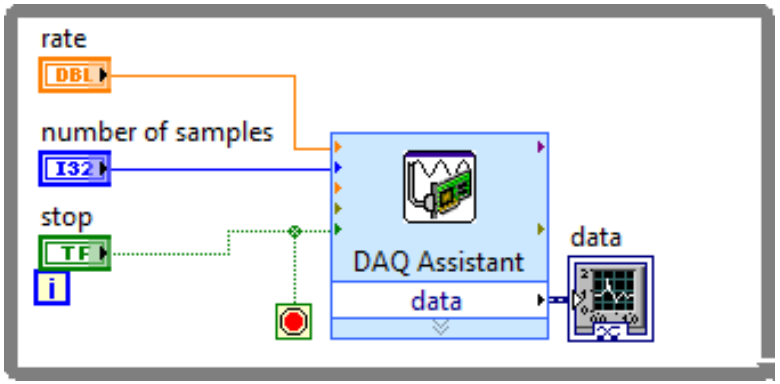
- Sličnan postupak i za digitalne/analogne izlaze.

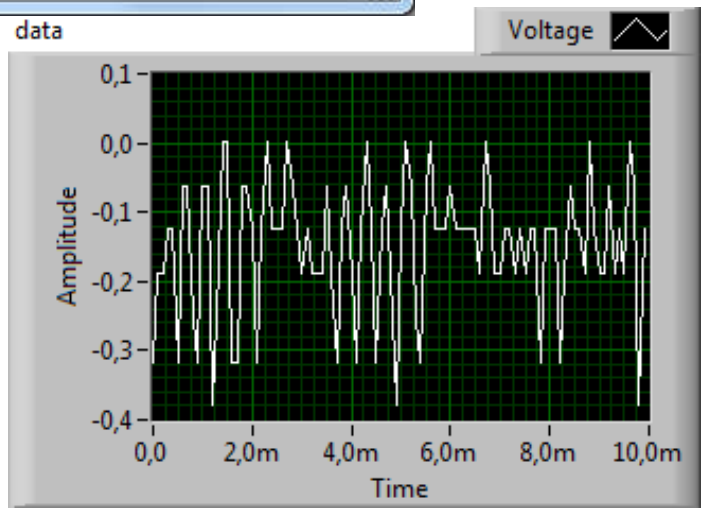
LabVIEW – *DAQ Assistant*

- *Measurement I/O » DAQmx – Data Acquistion* paleta.
- Ista procedura kreiranja *Task*-a kao i u *MAX*-u.



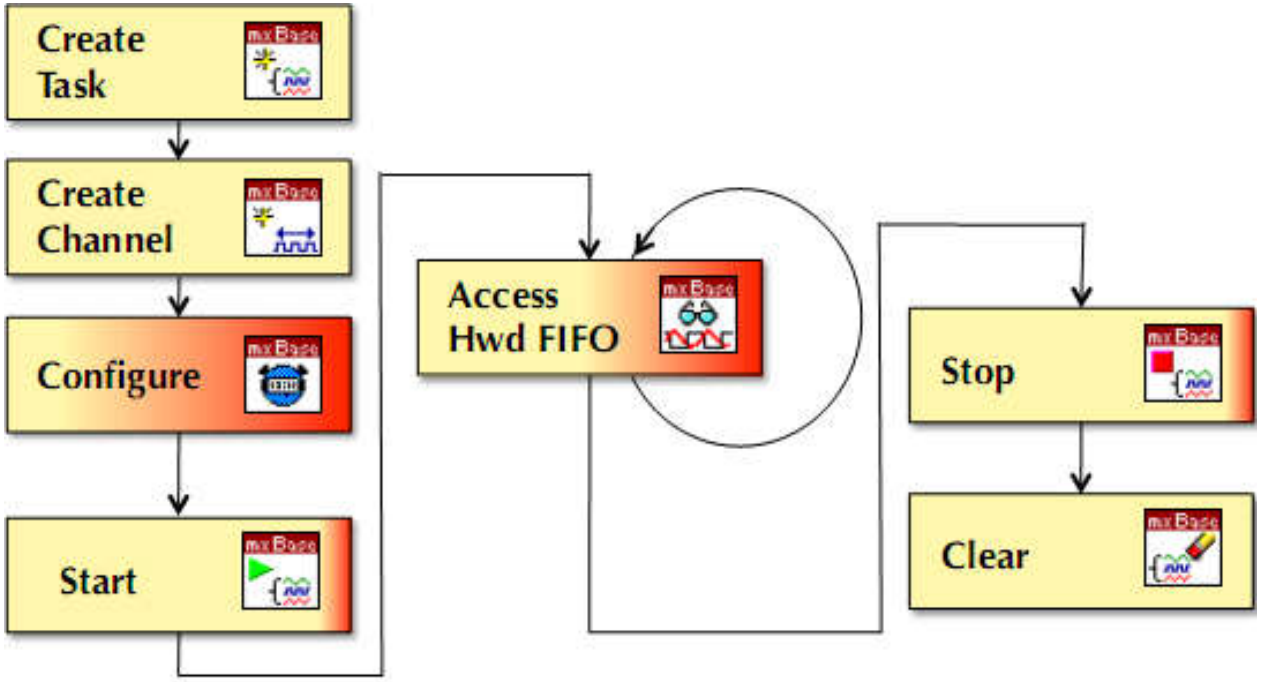






LabVIEW *DAQmx*

- Biblioteka funkcija za rad sa *NI DAQ* uređajima.
- Funkcije višeg nivoa, ali dovoljno detaljne za konfigurisanje najvećeg broja zadataka.
- Omogućava programiranje *DAQ* uređaja nezavisno od tipa urađaja i interfejasa (USB, PCI, PCIe, itd).
- Zamena *DAQ* uređaja drugim *DAQ* uređajem drugog tipa ne zahteva ili zahteva minimalnu promenu LV koda.
- *Measurement I/O » DAQmx – Data Acquistion* paleta.

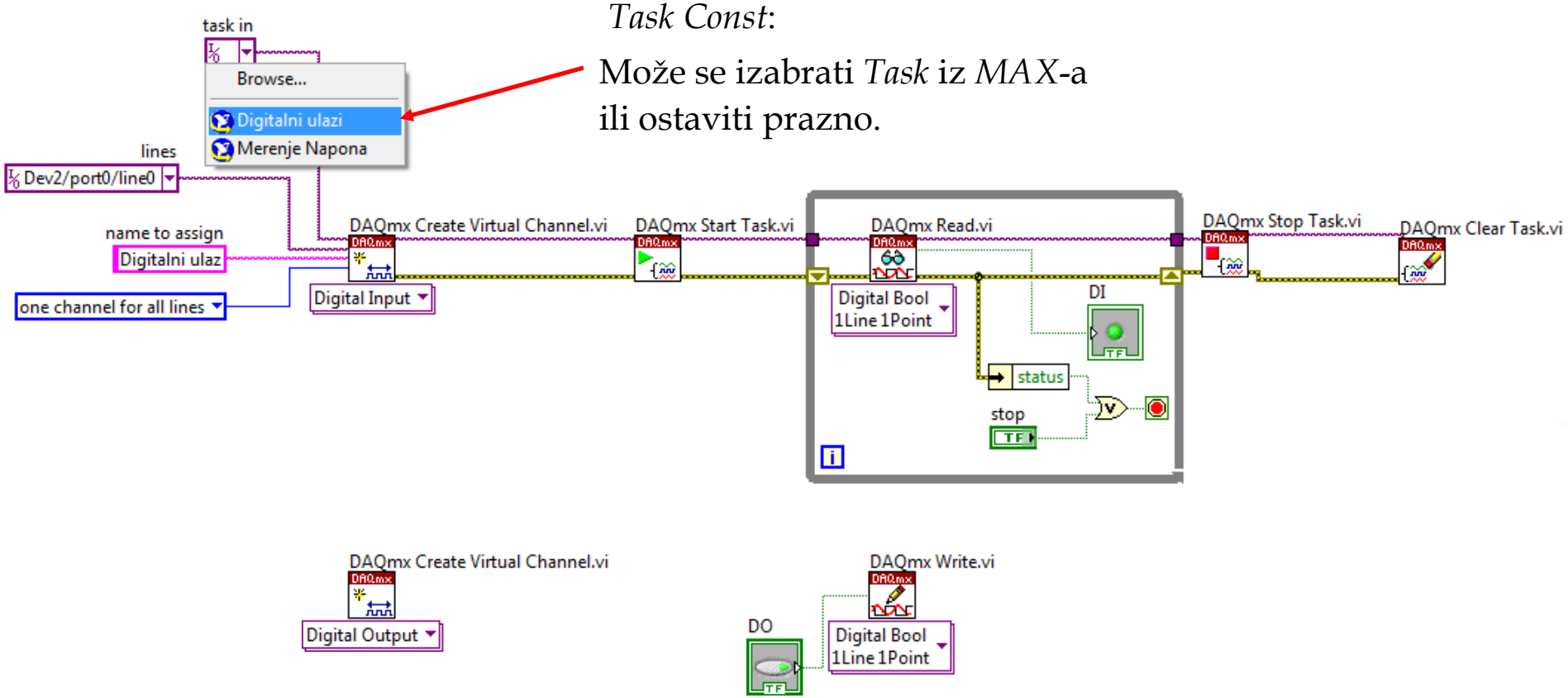


LabVIEW *DAQmx* – Digitalni I/O

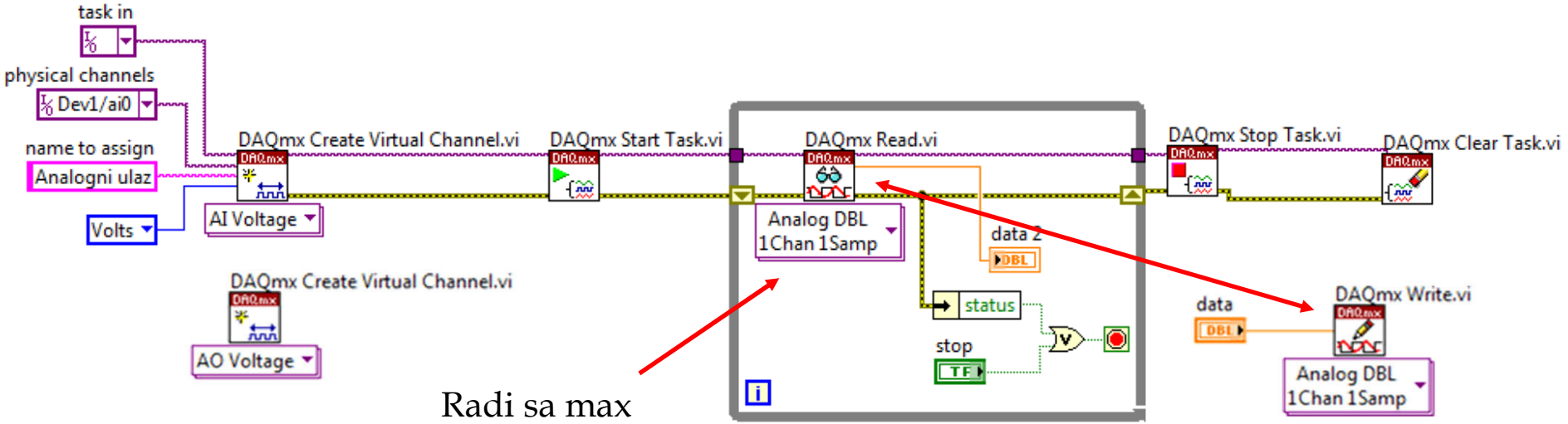
- *Virtual Channel* programski entitet koji uključuje fizički kanal (analogni I/O, digitalni I/O) zajedno sa informacijama koji su specifični za samo merenje kao što je opseg, skaliranje, kondicioniranje signali, itd.
- *Task* je kolekcija infromacija o jednom ili više kanala sa informacijama kao što je brzina akvizije, trigeri, itd.

Task Const:

Može se izabrati *Task* iz *MAX*-a ili ostaviti prazno.

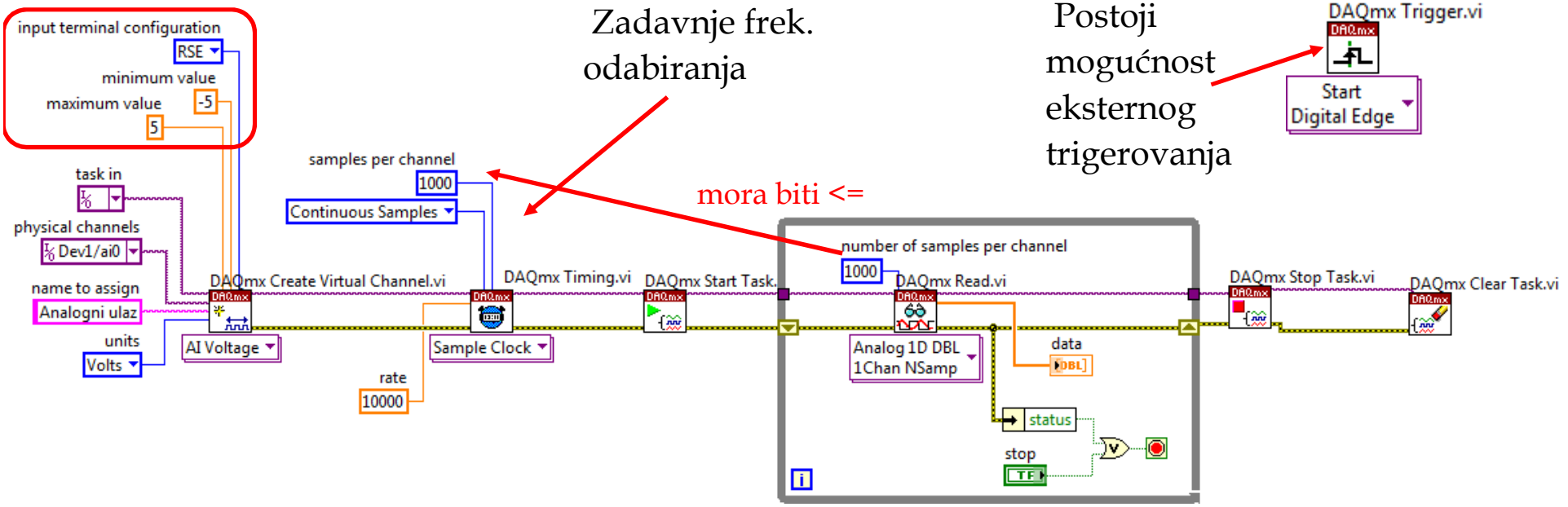


LabVIEW *DAQmx* – Analogni I/O



Radi sa max frek. odabiranja

Podešavanja



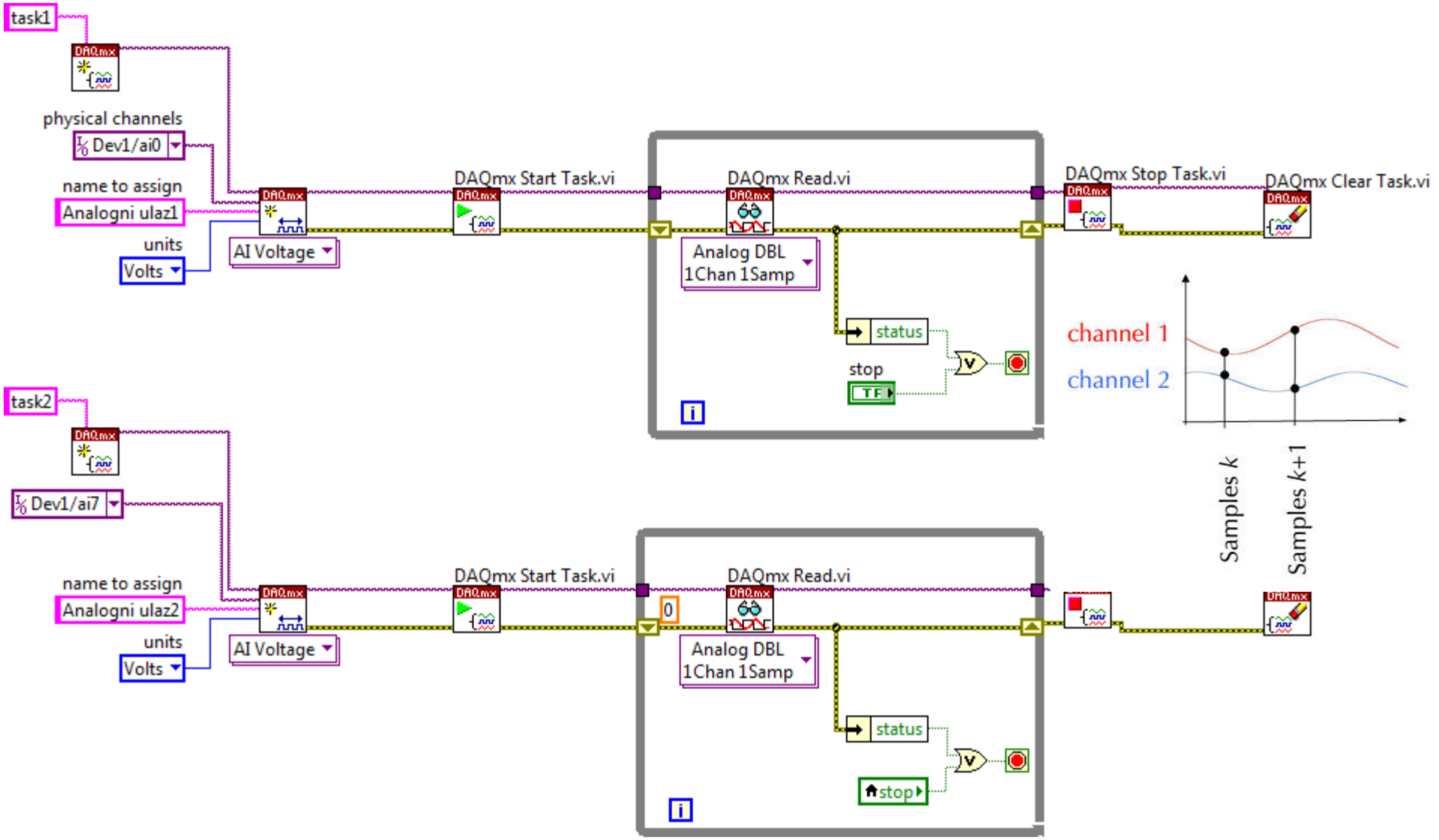
Zadavnje frek. odabiranja

mora biti <=

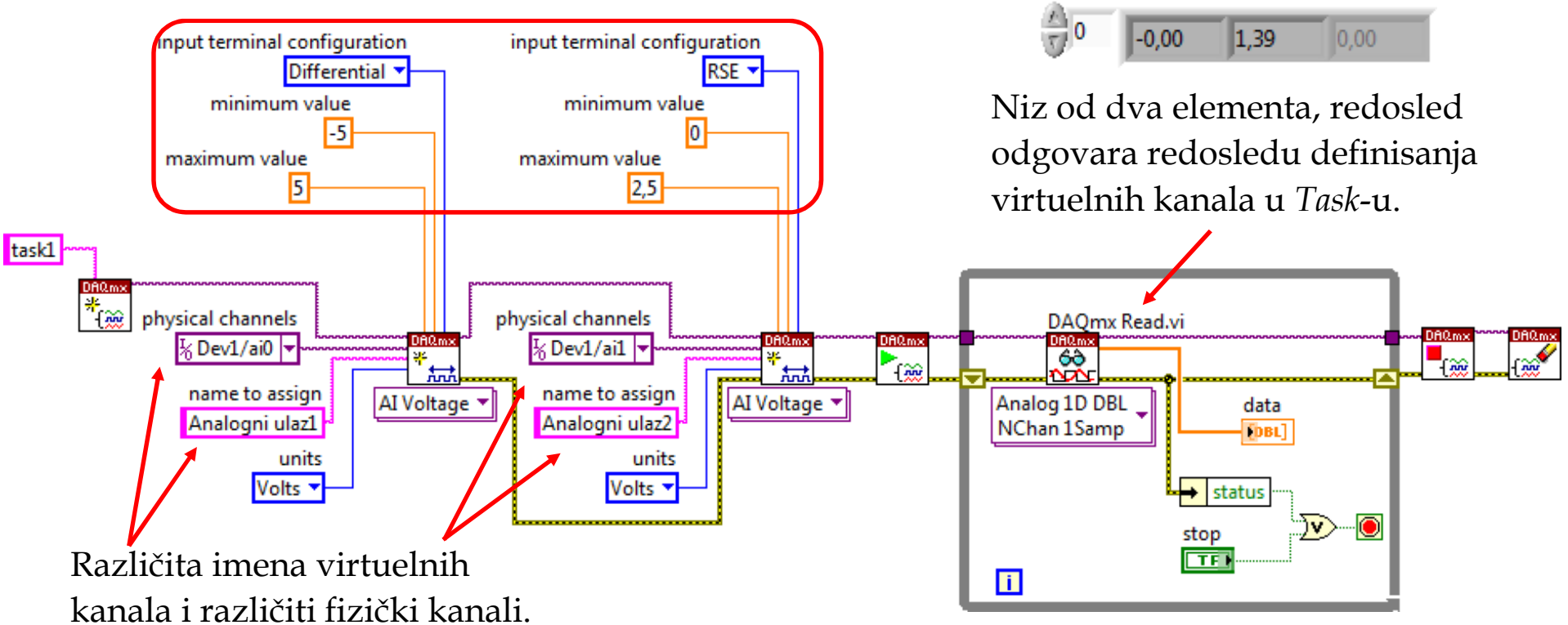
Postoji mogućnost eksternog trigerovanja

LabVIEW *DAQmx* – Analogni I/O

- Sledeći kod prijavljuje grešku pristupanja zauzetim resursima, iako su fizički kanali različiti kao i imena*Task*-ova. (Za digitalne *Task*-ove ovo ne važi.)



- Više analognih ulaza, ista frekvencija odabiranja.

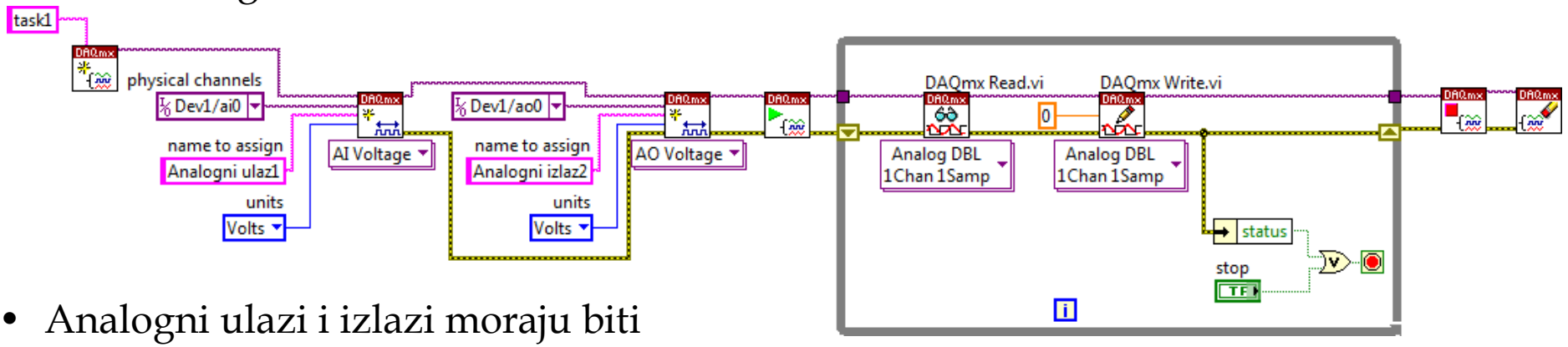


Različita imena virtuelnih kanala i različiti fizički kanali.

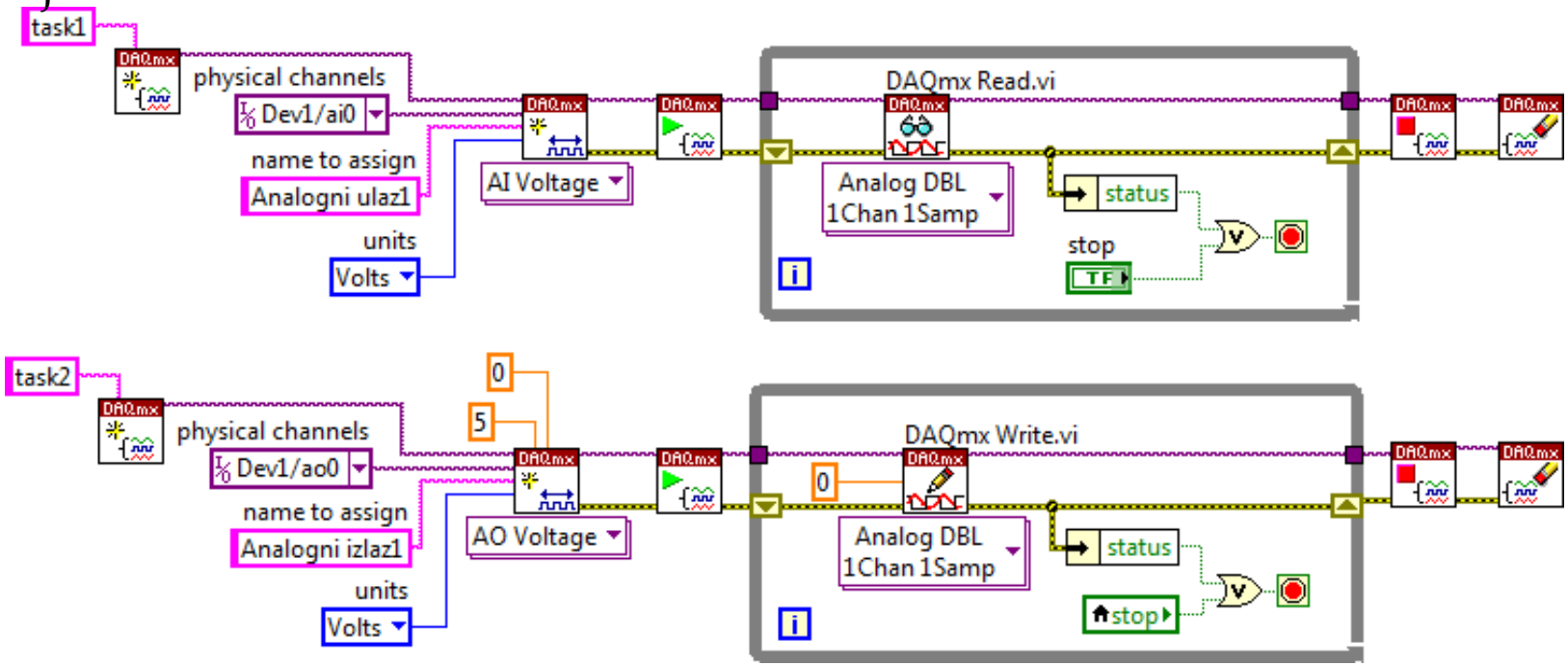
Niz od dva elementa, redosled odgovara redosledu definisanja virtuelnih kanala u *Task*-u.

LabVIEW *DAQmx* – Analogni I/O

- Sledeći kod javlja grešku, jer se ne mogu fizički kanali različitih funkcija definisati u okviru istog *Task*-a.

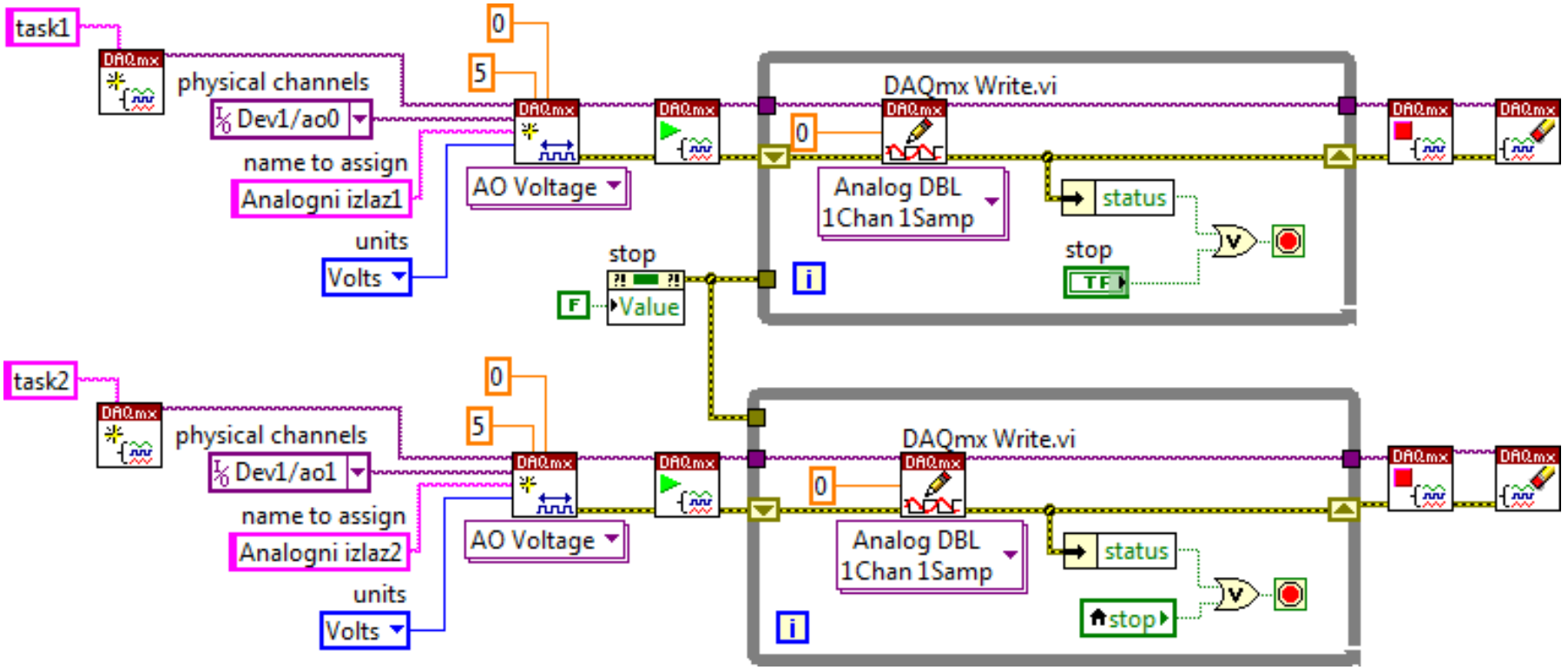


- Analogni ulazi i izlazi moraju biti dodeljeni različitim *Task*-ovima.



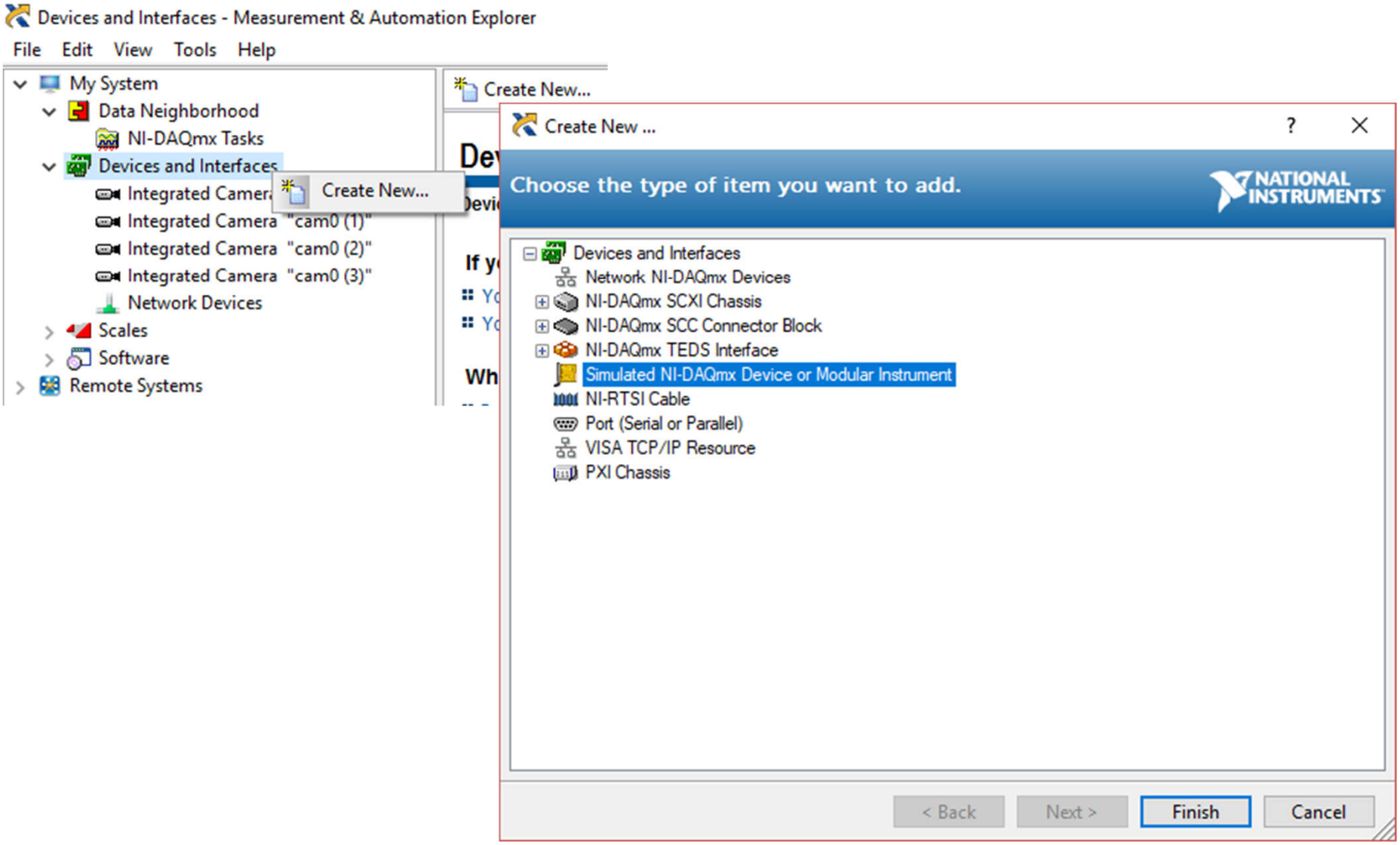
LabVIEW *DAQmx* – Analogni I/O

- Različiti analogni izlazi mogu biti dodeljni različitim *Task*-ovima i neće biti prijavljena greška.

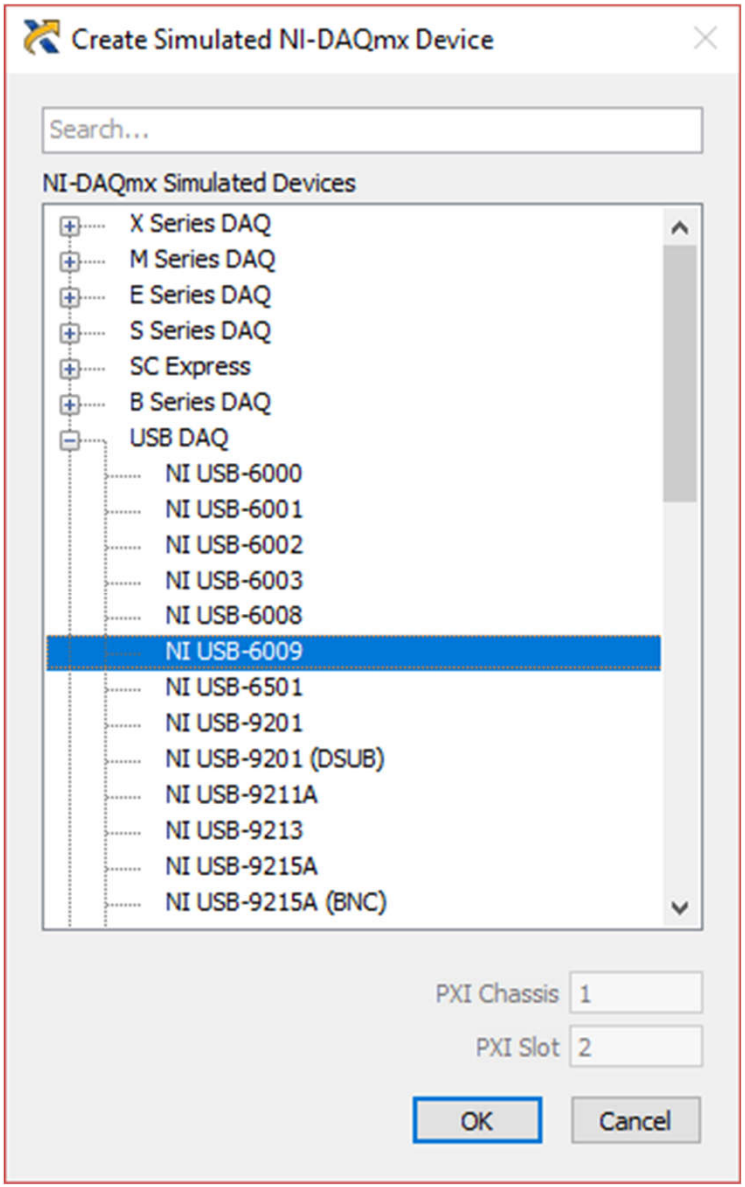


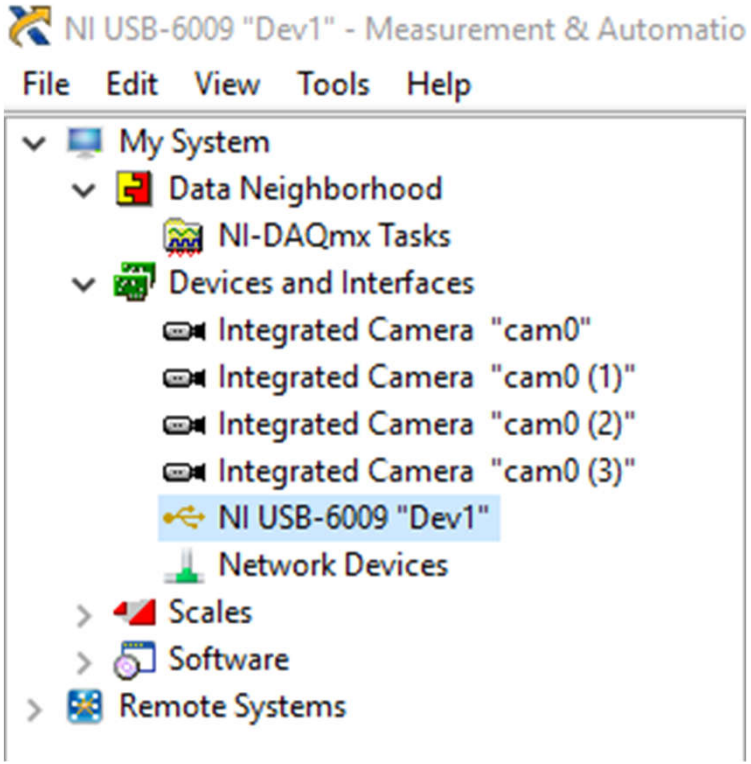
LabVIEW *DAQmx* – Simulirana kartica

- U toku razvoja možda nije uvek dostupna realna kartica. LabVIEW ostavlja mogućnost kreiranja simulirane kartice.



LabVIEW *DAQmx* – Simulirana kartica

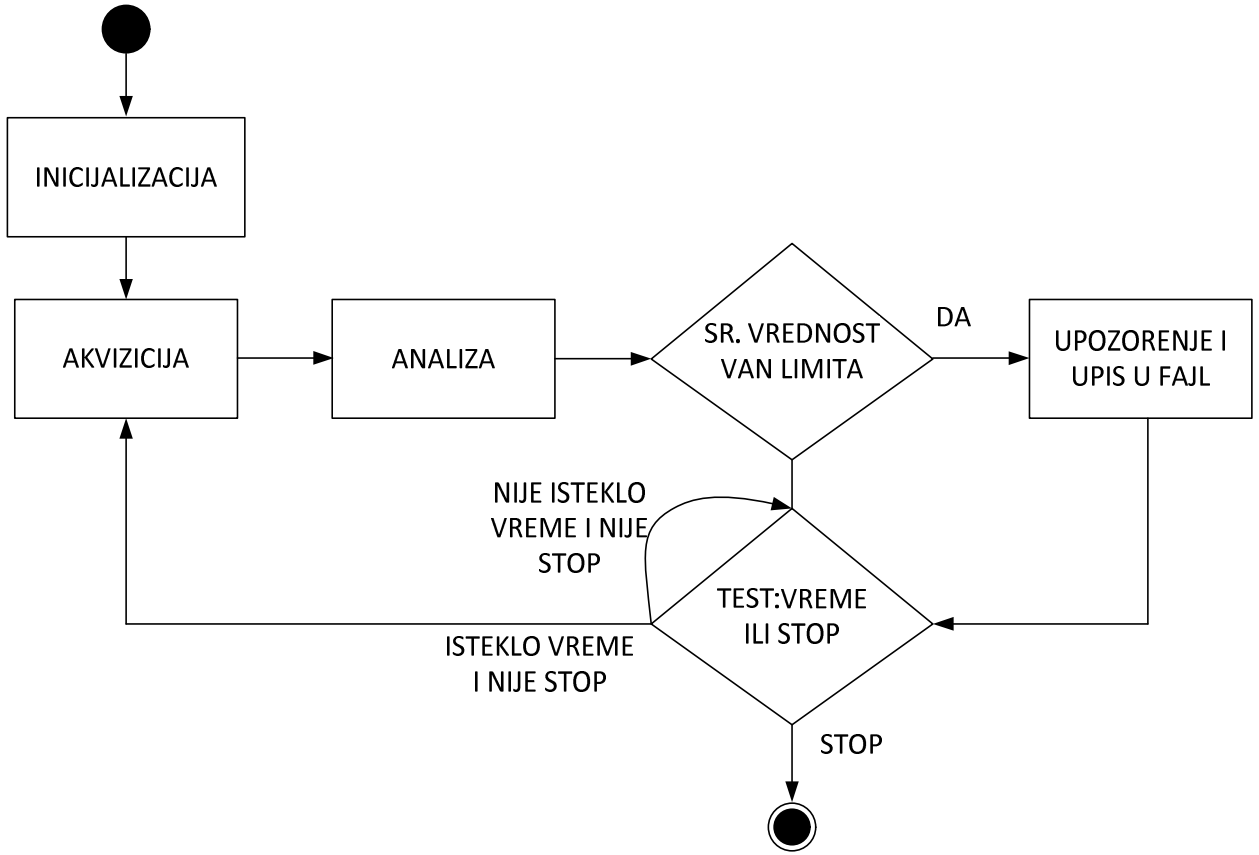




- U LabVIEW se "vidi" kao i svaka druga kartica.

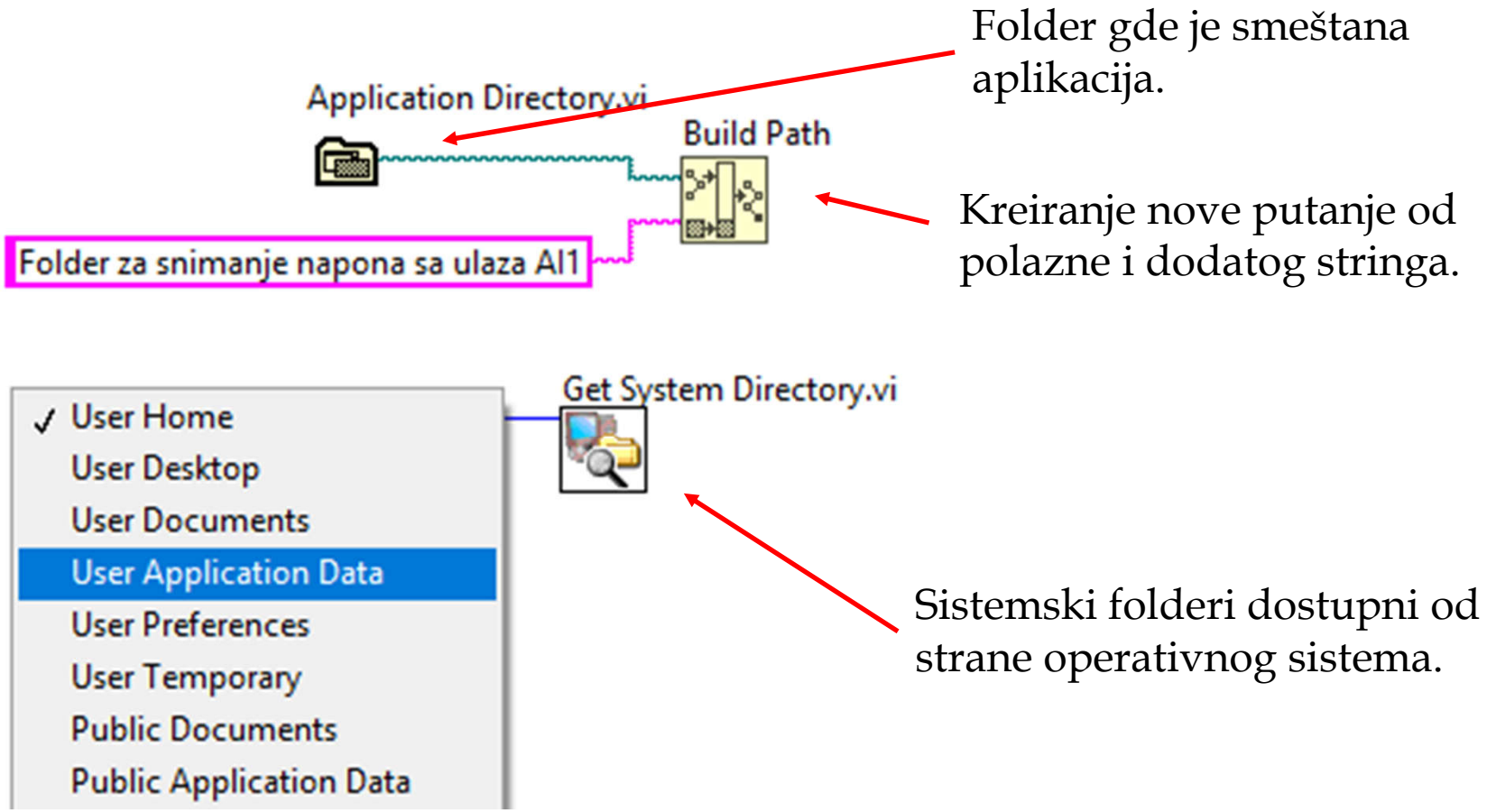
Vežba 18

- Nadogradnja prethodnog zadatka. Umesto simuliranog signala koristiti realan analogni signal (dobijen pomoću potenciometra). Pretpostaviti da je to nivo vode u rezervoaru. Ukoliko je signal veći od maksimalno dozvoljene visine aktivirati odgovarajuću LED, koja predstavlja indikaciju alarm (ALH – *alarm level high*). Slično obezbediti i za donji granični nivo (ALL – *alarm level low*). Dodati i sledeći funkcionalnost: alarm se gasi po vraćanju nivoa u dozvoljeni opseg i aktiviranje hardverskog ACK (*acknowledgement*) koji je potrebno obezbediti kao digitalni ulaz.



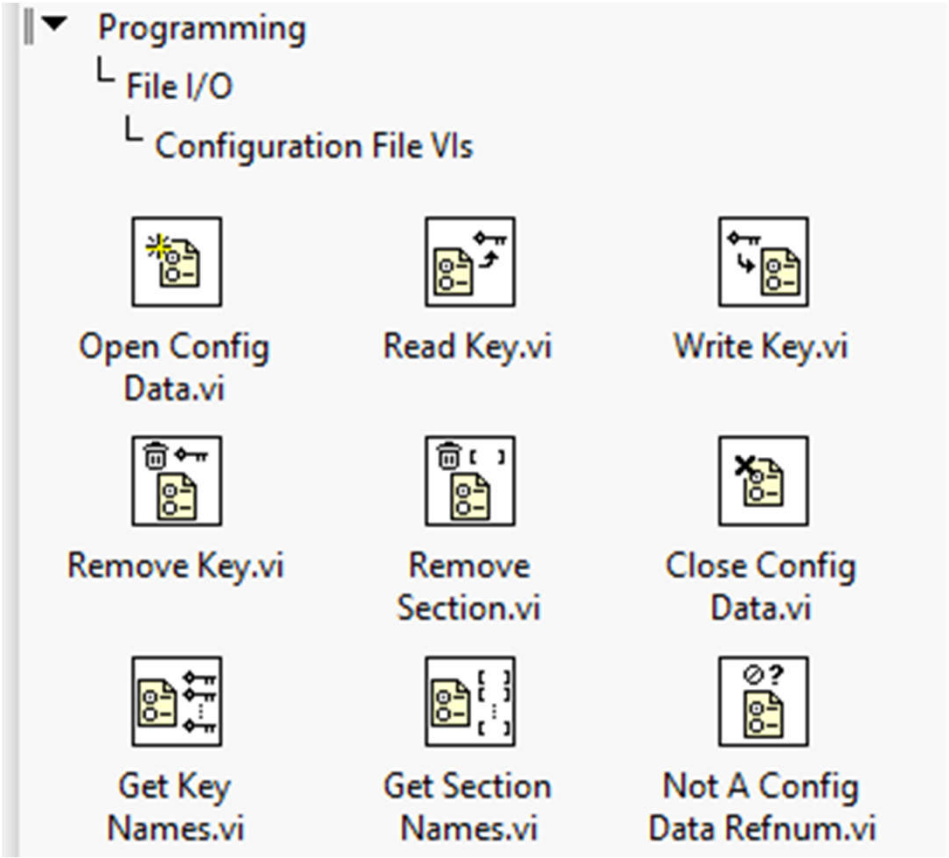
Priprema projekta za distribuciju

- VI *Properties* – proveriti da li je potrebno dozvoliti/zabraniti minimizovanje VI, zatvaranje na X, dozvola korišćenja *toolbox*-a, *scrollbar*-a, promena dimenzija VI i drugo.
- Putanje – obezbediti putanje za snimanje podataka, učitavanje parametara potrebnih za inicijalizaciju, korišćenje standardnih foldera (aplikacioni, *user* i slično)



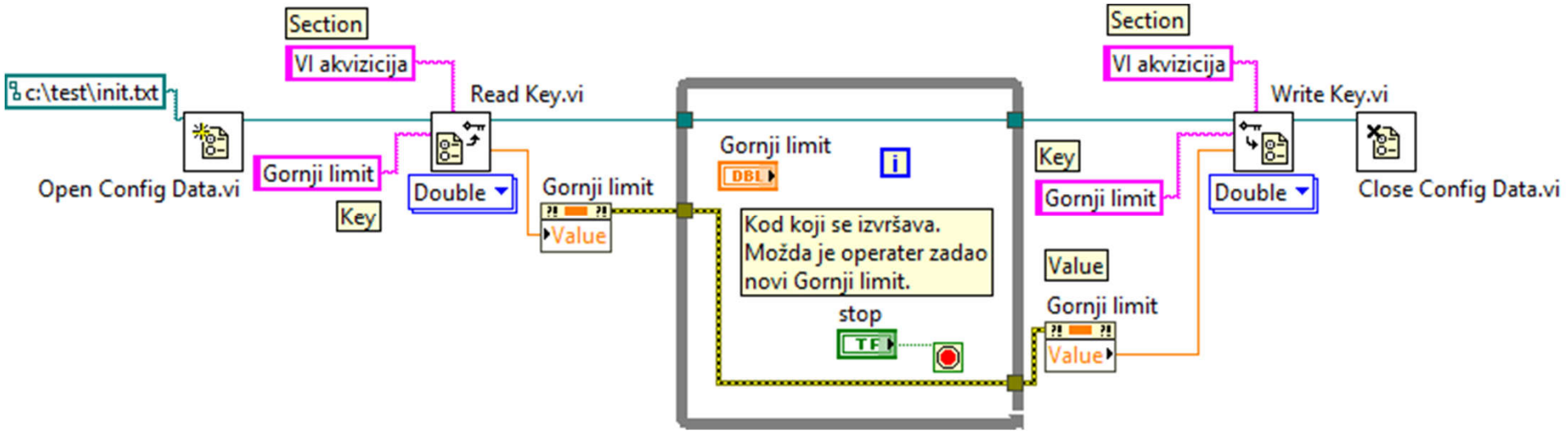
Priprema projekta za distribuciju

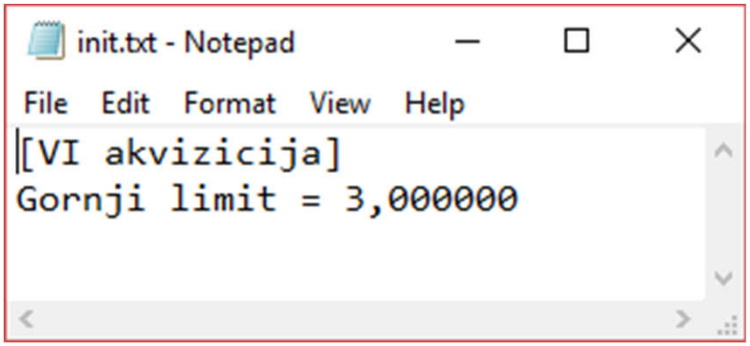
- Inicijalizacija parametara.
- Paleta za formiranje i rada sa konfiguracionim fajlovima.



Priprema projekta za distribuciju

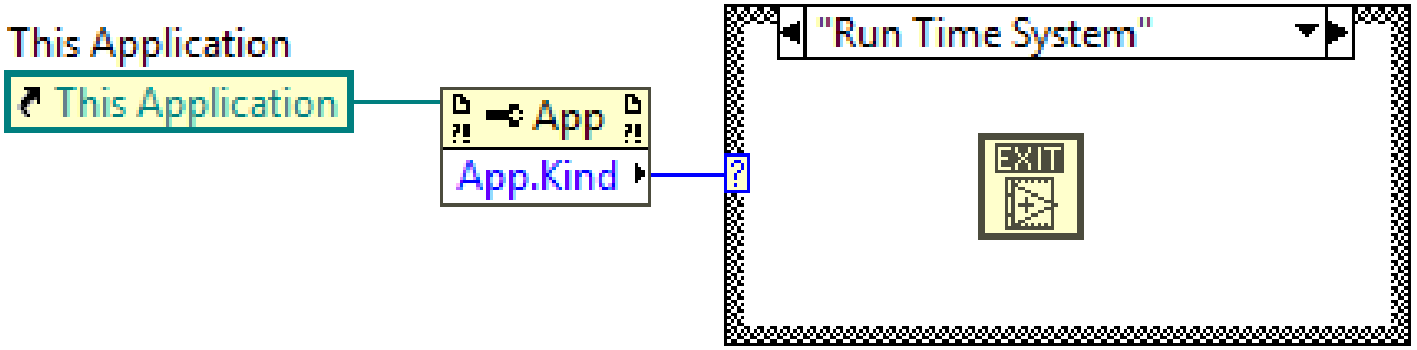
- Primer učitavanja i snimanja vrednosti pomoću konfiguracionog fajla.



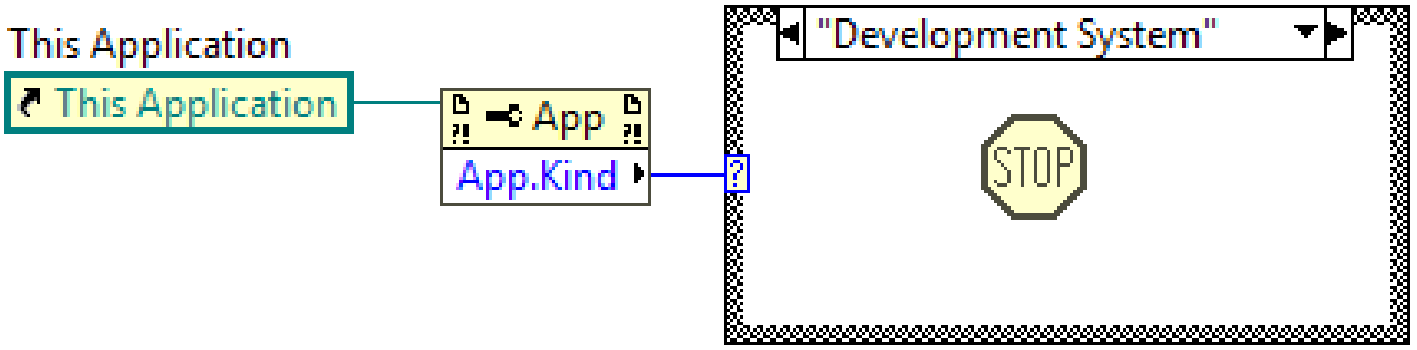


Priprema projekta za distribuciju

- Izlazak iz aplikacije – potrebno je obezbediti da se ugasi i *Run Time* modul.

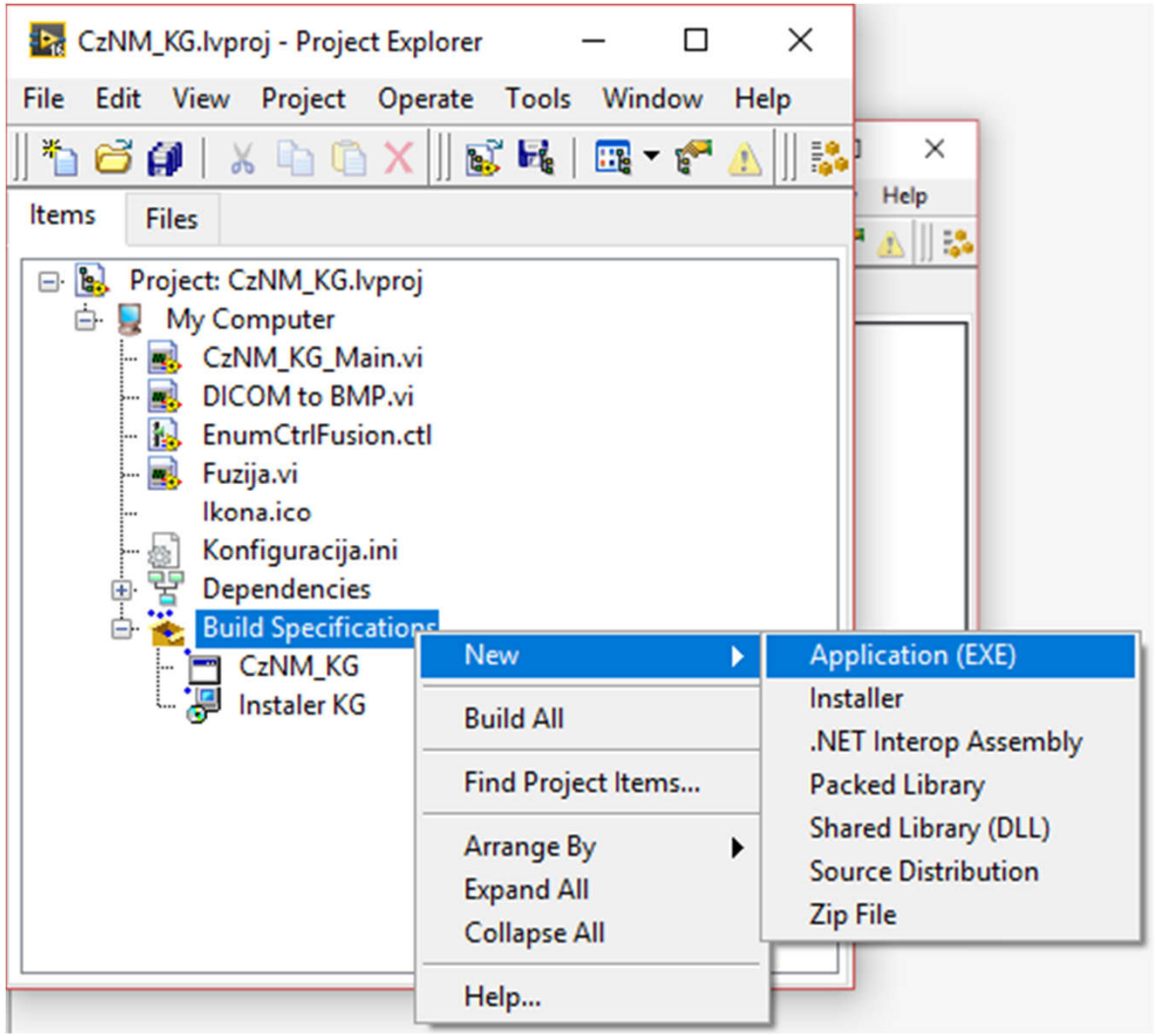


- Obezbeđivanje da se ne ugasi svaki put LabVIEW pri zaustavljanju VI-a u toku samog razvoja aplikacije



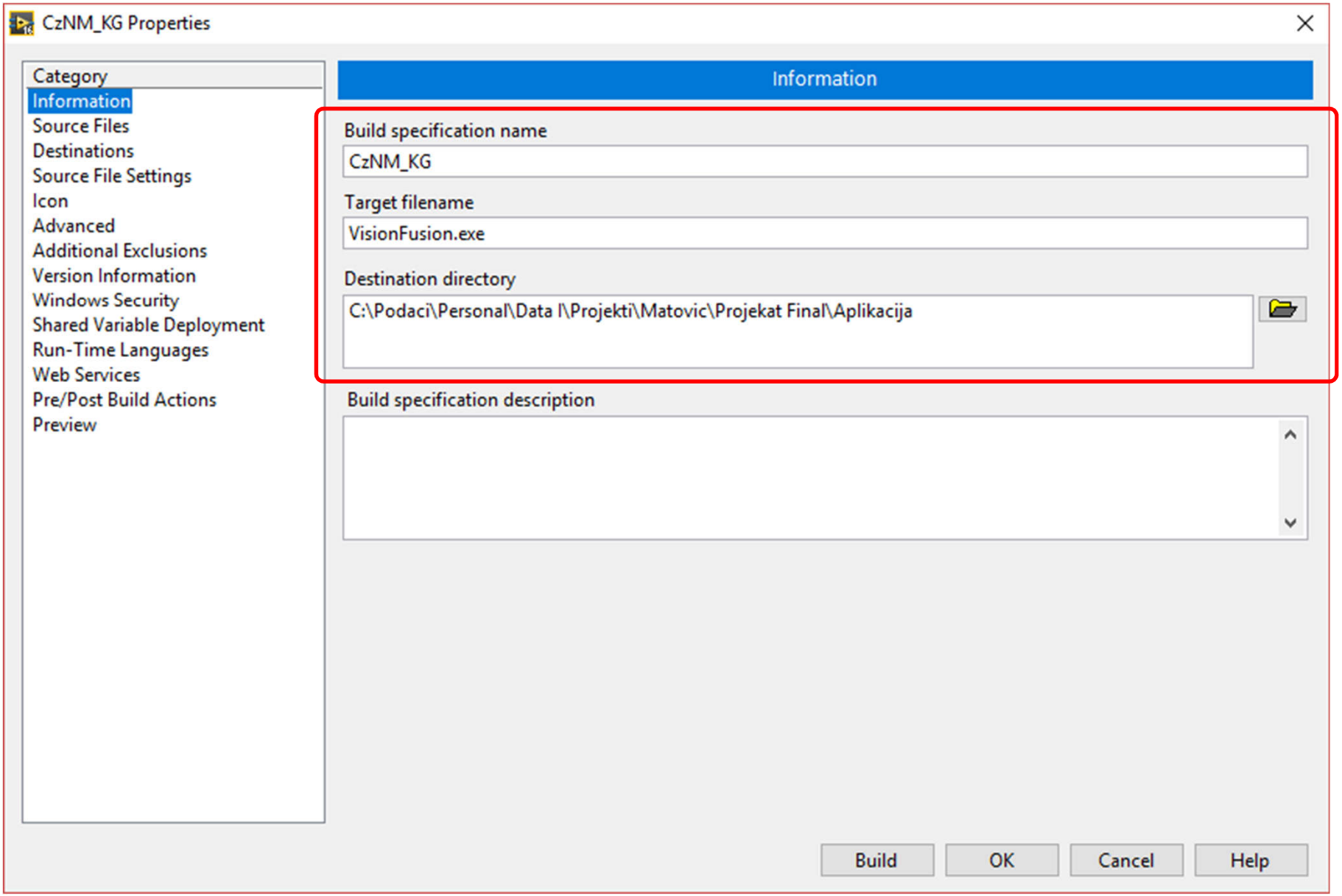
Kreiranje .exe fajla

- Prvo je potrebno kreirati izvršni fajl na osnovu glavnom VI-a, tj. VI-a koji pokreće celo rešenje.
- Iz *Project Explorer*-a izabrati **Build Specification – New – Application (EXE)**.



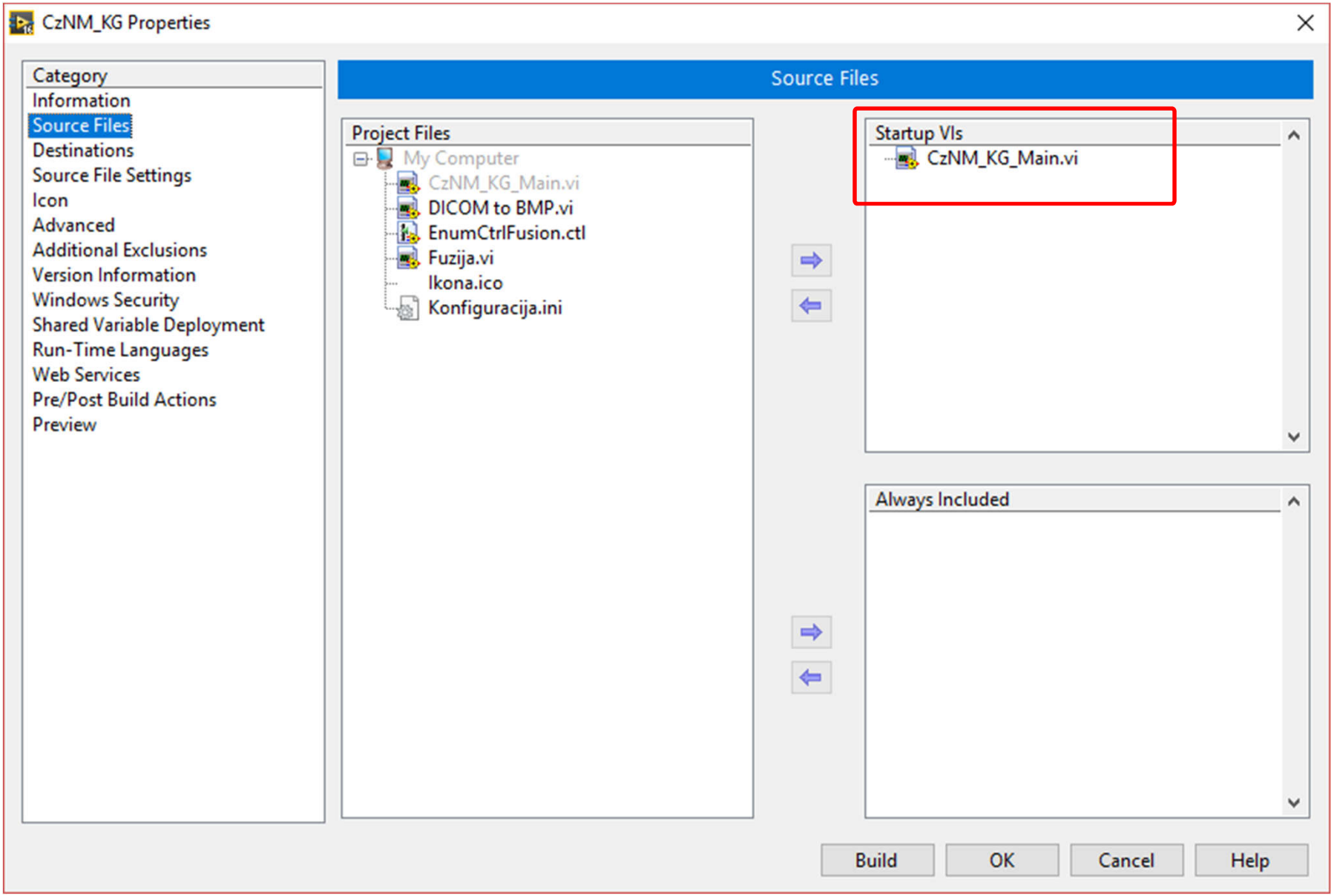
Kreiranje .exe fajla

- Definisanje osnovnih informacija, naziva EXE i putanje EXE fajla (nakon kreiranja).



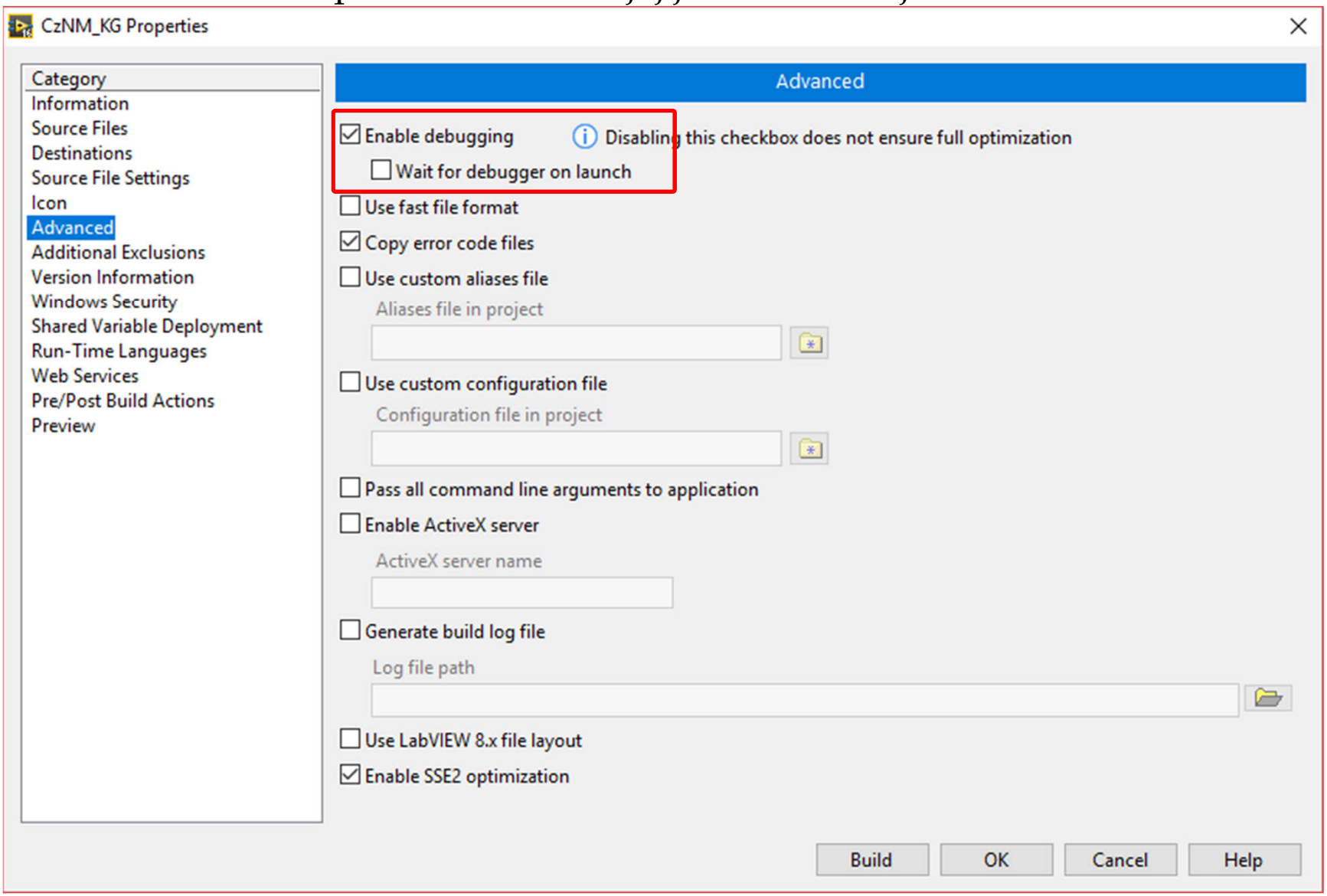
Kreiranje .exe fajla

- Definisanje **glavnog VI**.



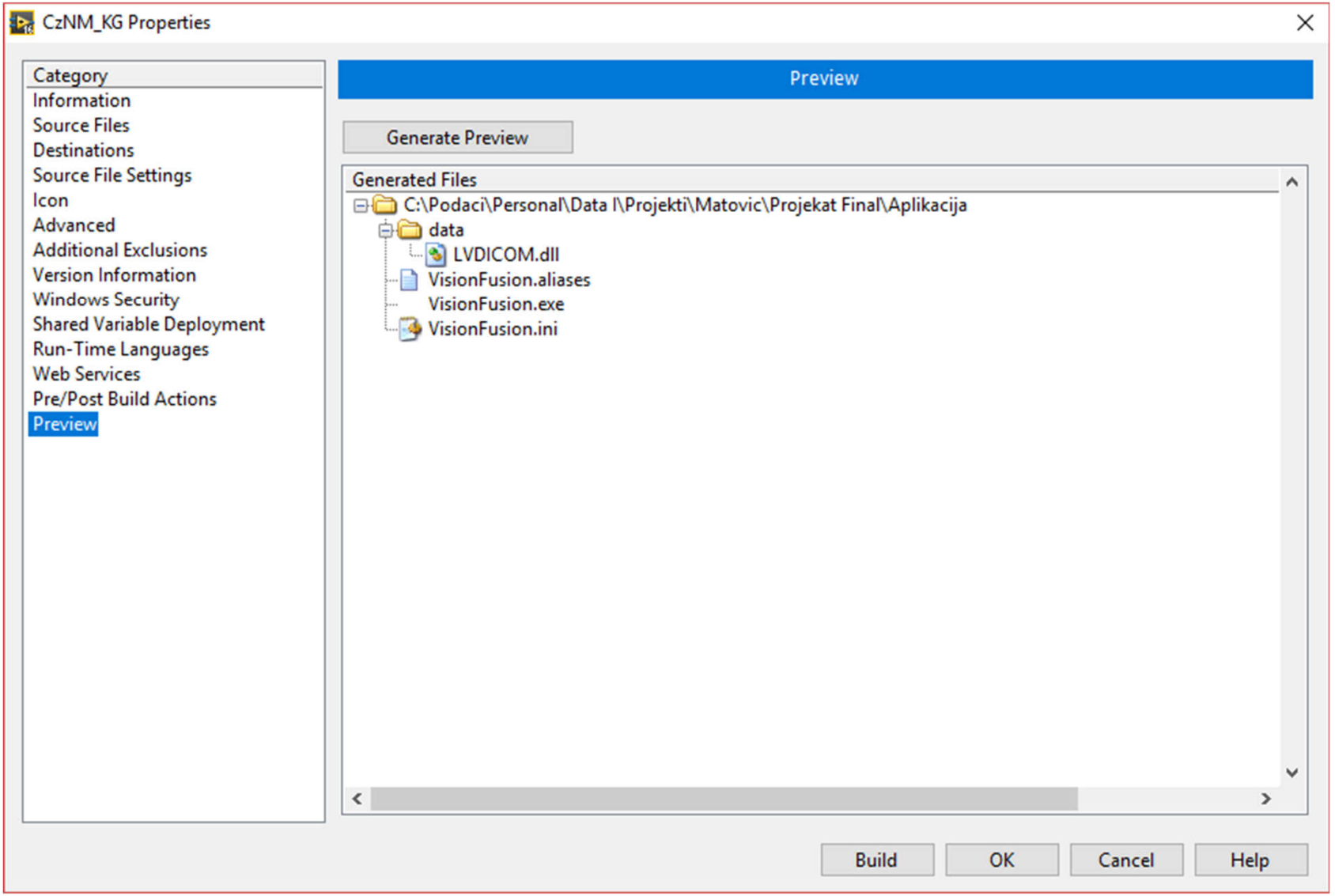
Kreiranje .exe fajla

- Ostaviti mogućnost debagiranja i na *target*-u. Bitno ako se jave problemi zbog hardverske razlike sa platformom na kojoj je vršen razvoj.

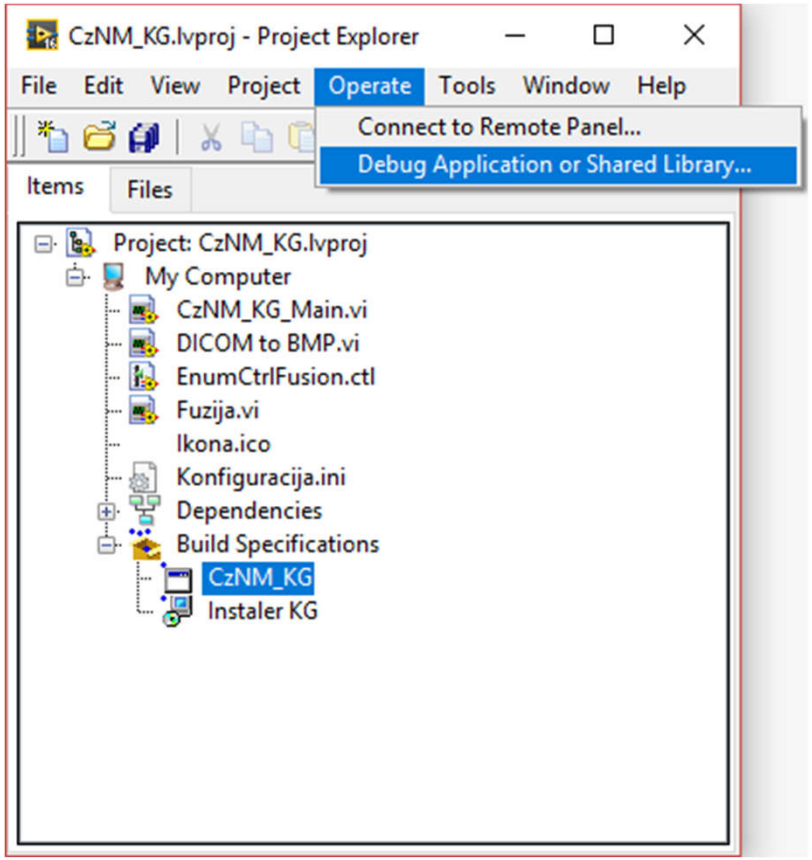


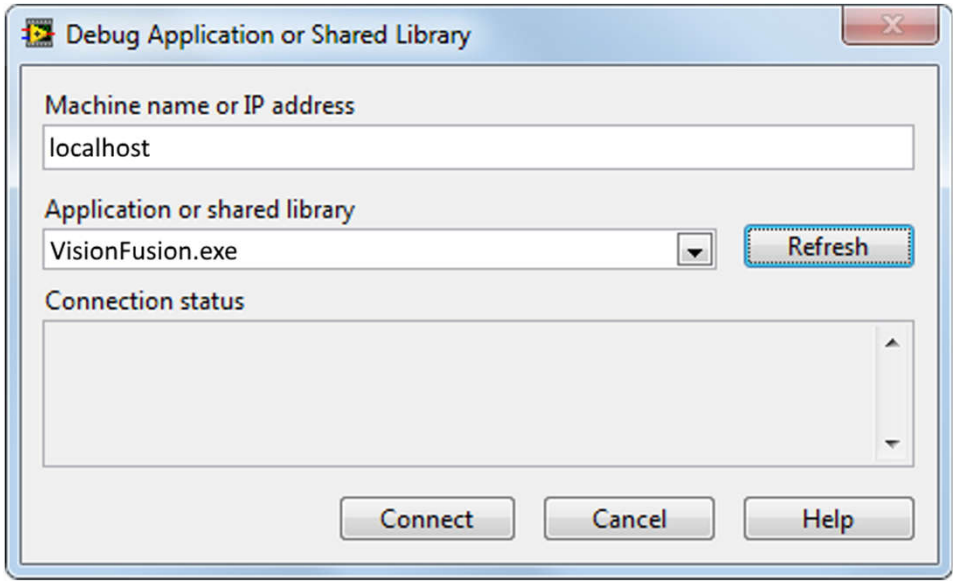
Kreiranje .exe fajla

- Proveriti strukturu finalnog direktorijuma.
- Prvo **OK** (sledi pitanje za **SAVE**), pa tek onda **Build**.



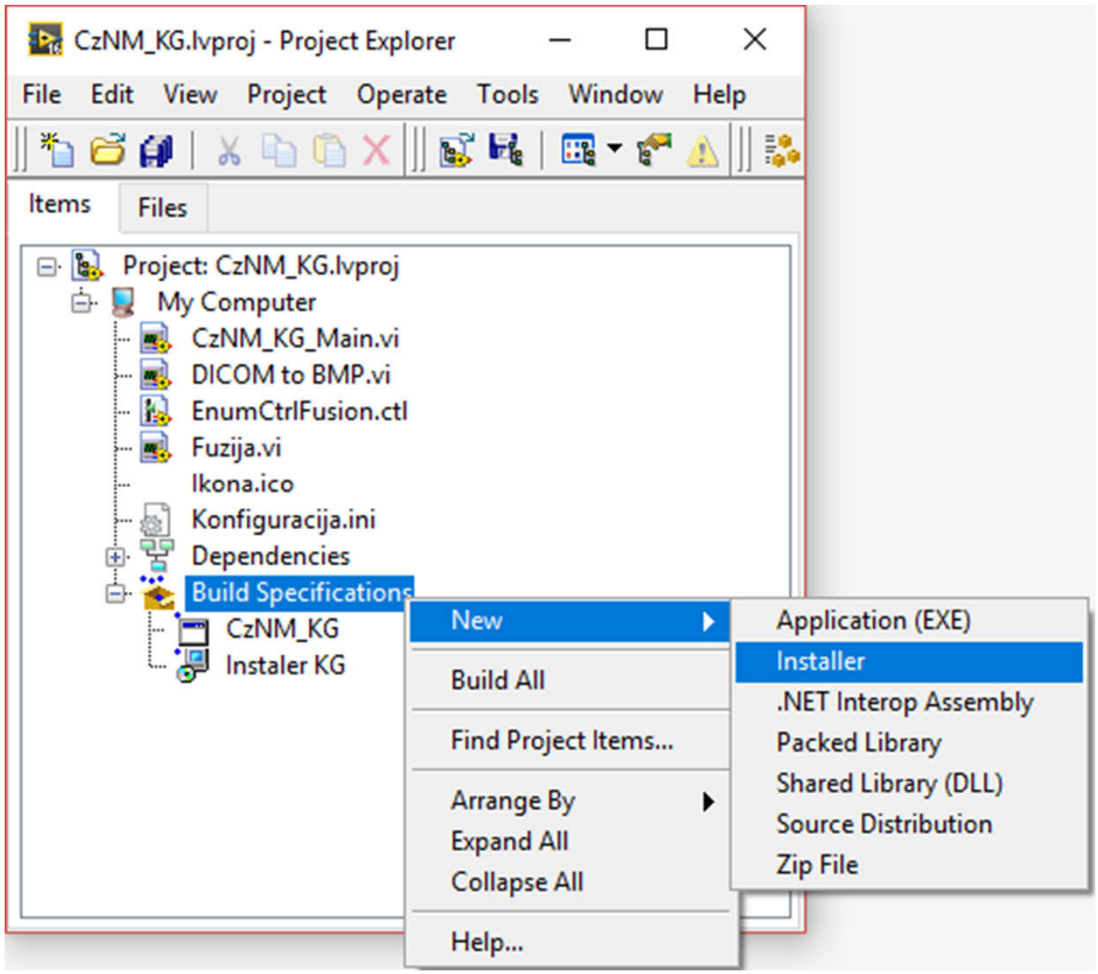
Pristup .exe fajlu na target-u radi debagiranja





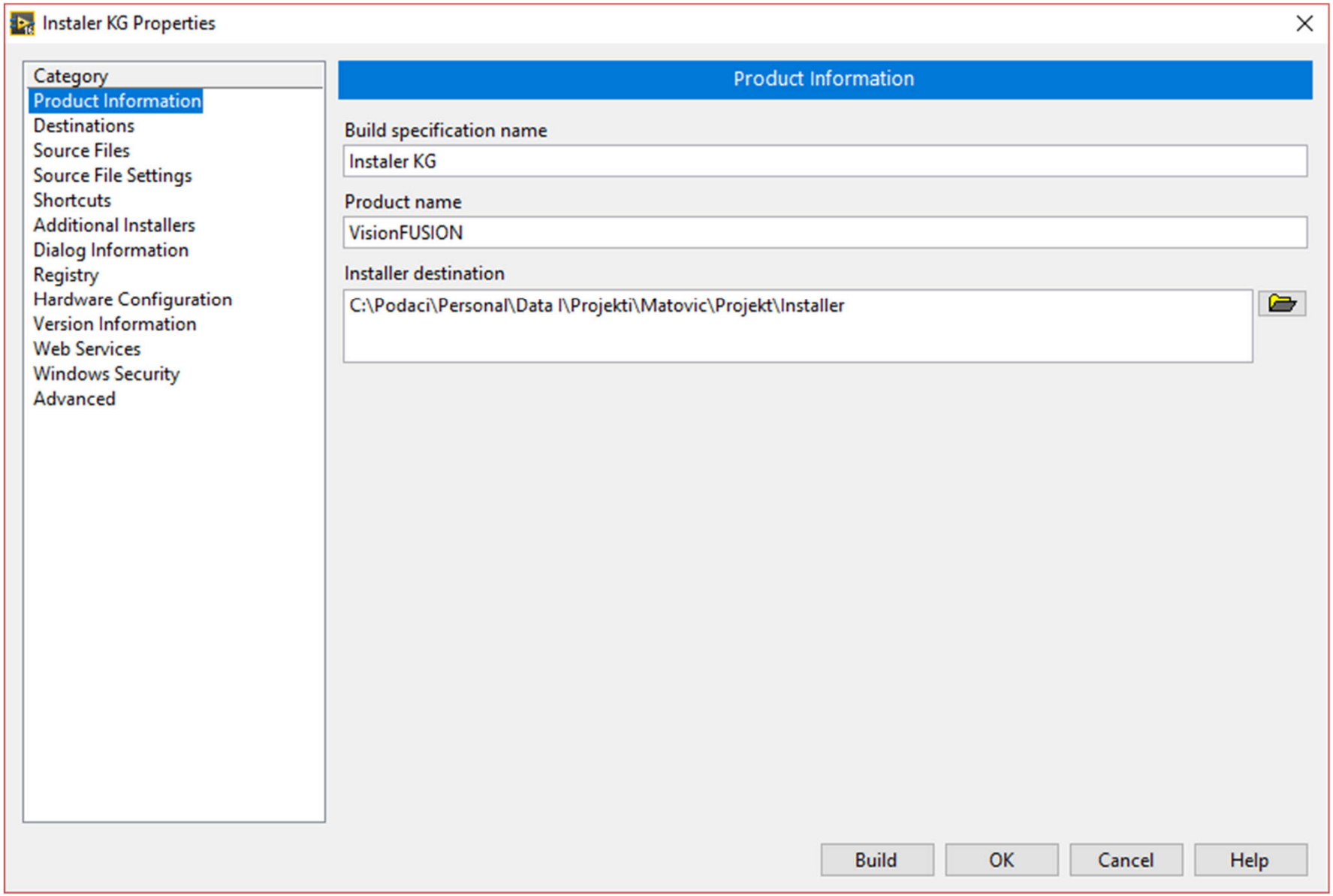
Kreiranje *installer*-a

- Na *target*-u nije potreban ceo LabVIEW paket (bespotrebno plaćanje licenci).
- *Installer* sadrži samo neophodne komponente za rad finalne aplikacije.
- Iz *Project Explorer*-a izabrati **Build Specification – New – Installer**.



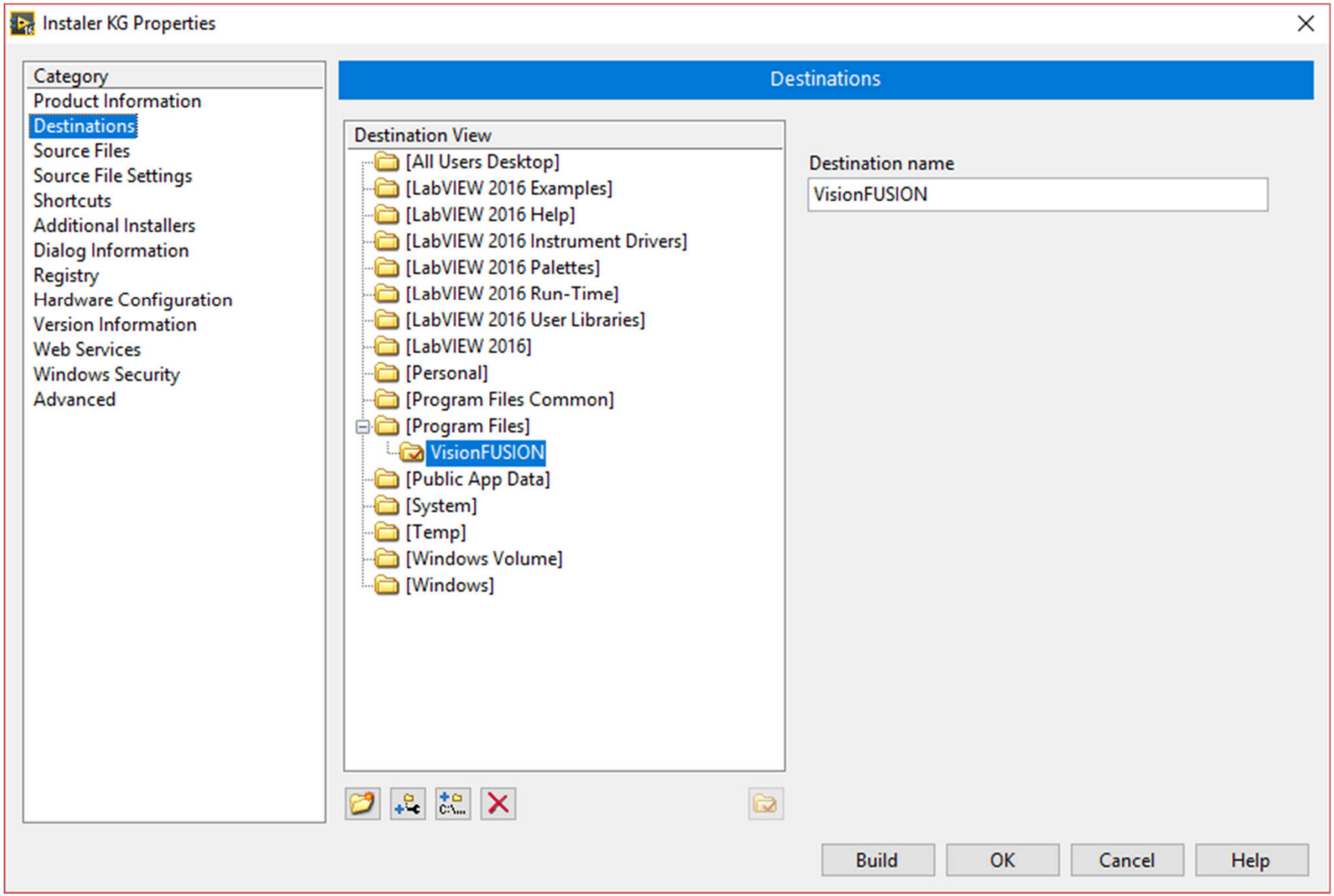
Kreiranje *installer*-a

- Ime installer-a i gde se nalazi (na računaru na kome se i kreira).



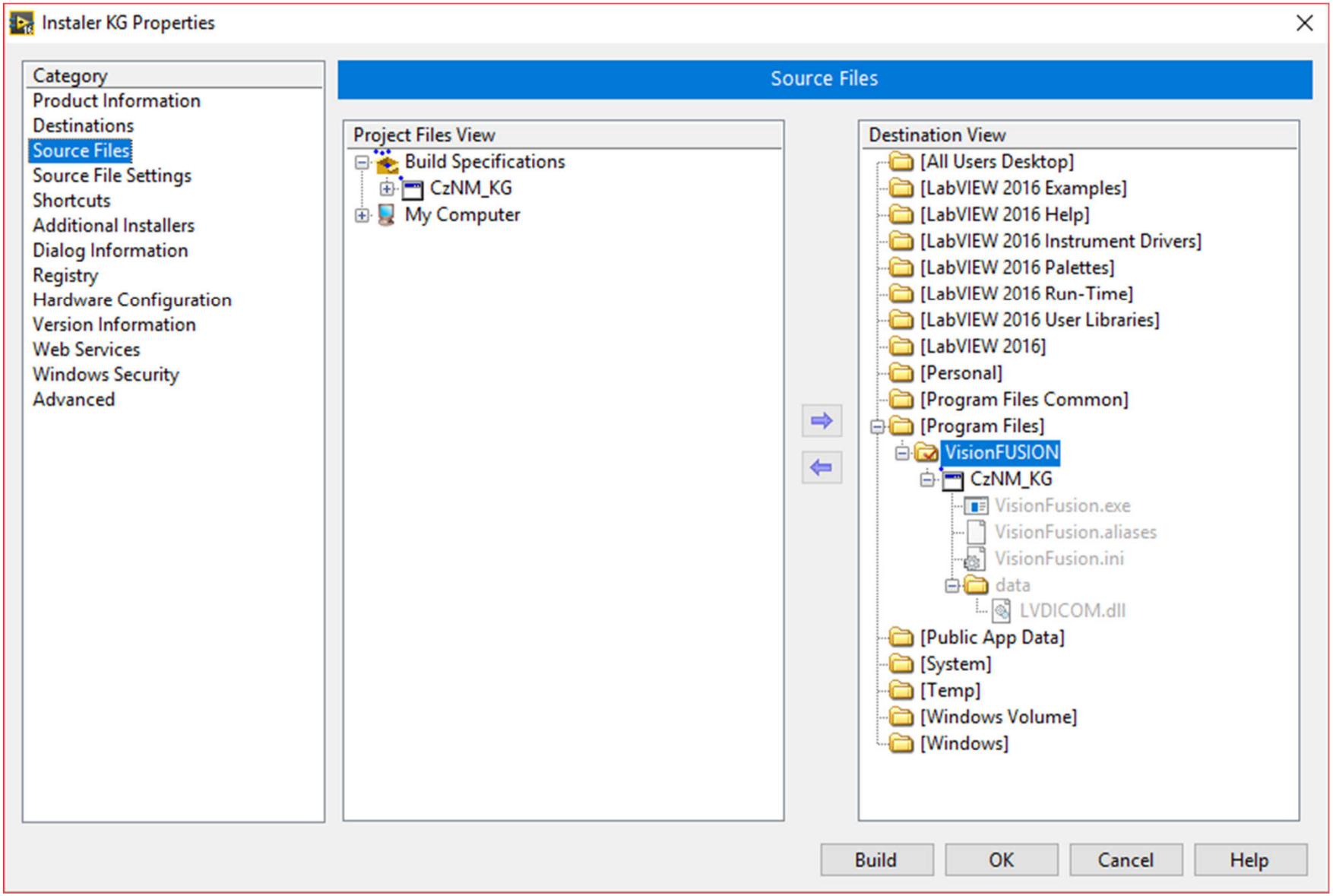
Kreiranje *installer*-a

- Destinacija svih fajlova koje proizvede *intaller* na *target*-u.



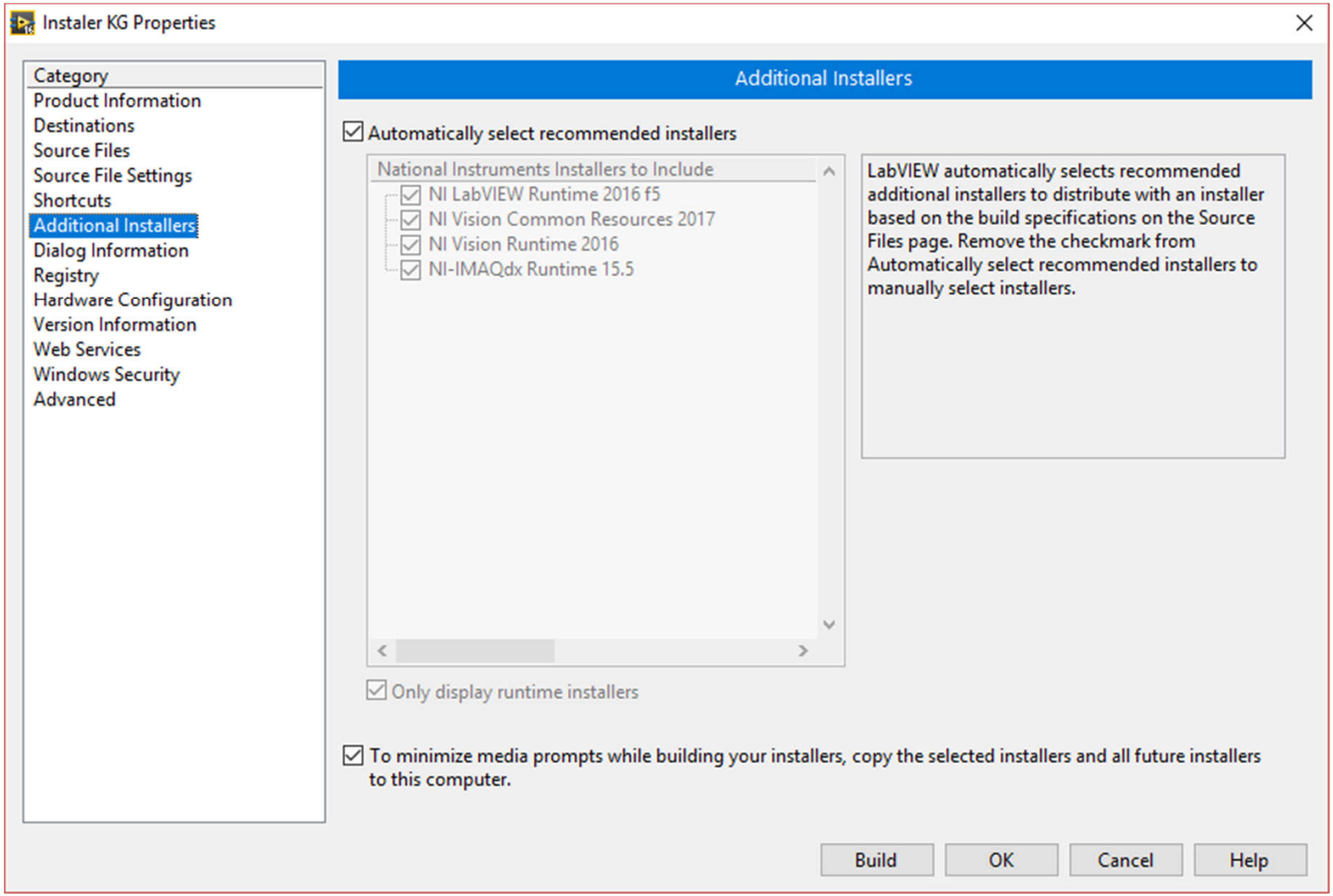
Kreiranje *installer*-a

- Izvorni fajl (već kreirani .exe) za koji se pravi *installer*.



Kreiranje *installer*-a

- Dodavanje dodatnih *installer*-a po potrebi. Na slici su to **NI Vision Runtime 2016** i **Ni-IMAXdx Runtime 15.5**.



Kreiranje *installer*-a

- Dodatne informacije – verzija, kompanija koja je napravila aplikaciju, itd.