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

Practicum of measurement and data acquisition systems

Using loops and decision-making structures

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

Itasdi

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

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

Praktikum iz merno-akvizicionih sistema

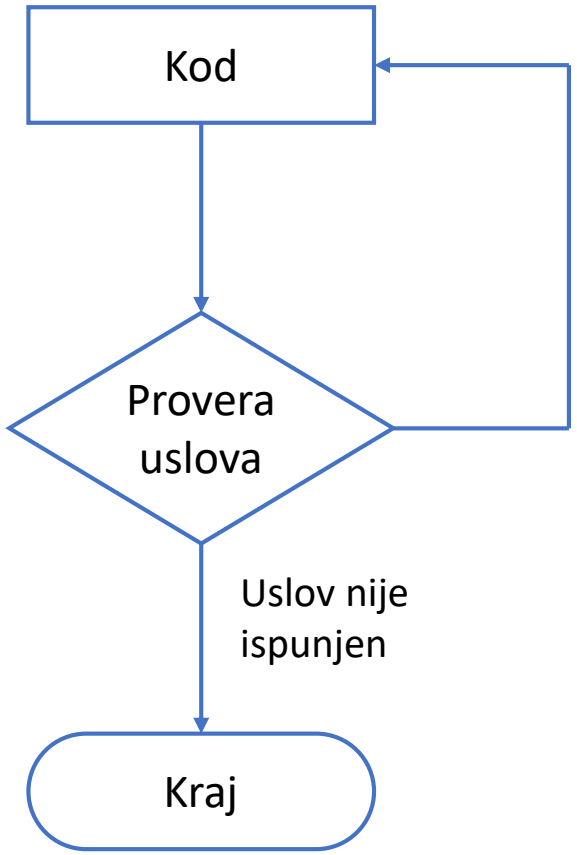
Korišćenje petlji i struktura za odlučivanje

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

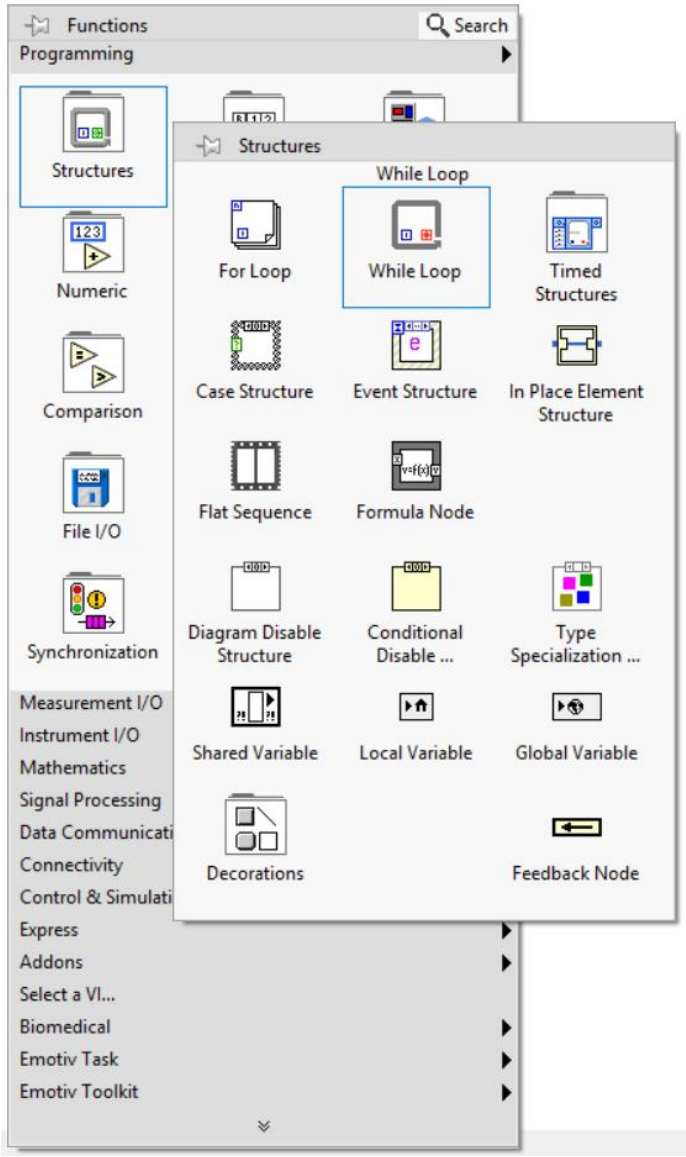
Itasdi

While petlja

Kod se izvršava unutar while petlje sve dok se ne dogodi specifičan uslov.

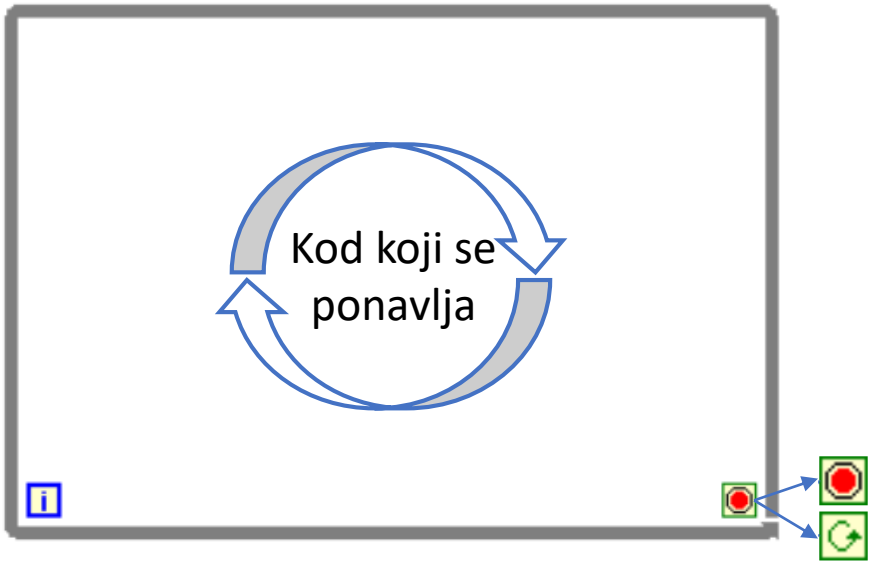


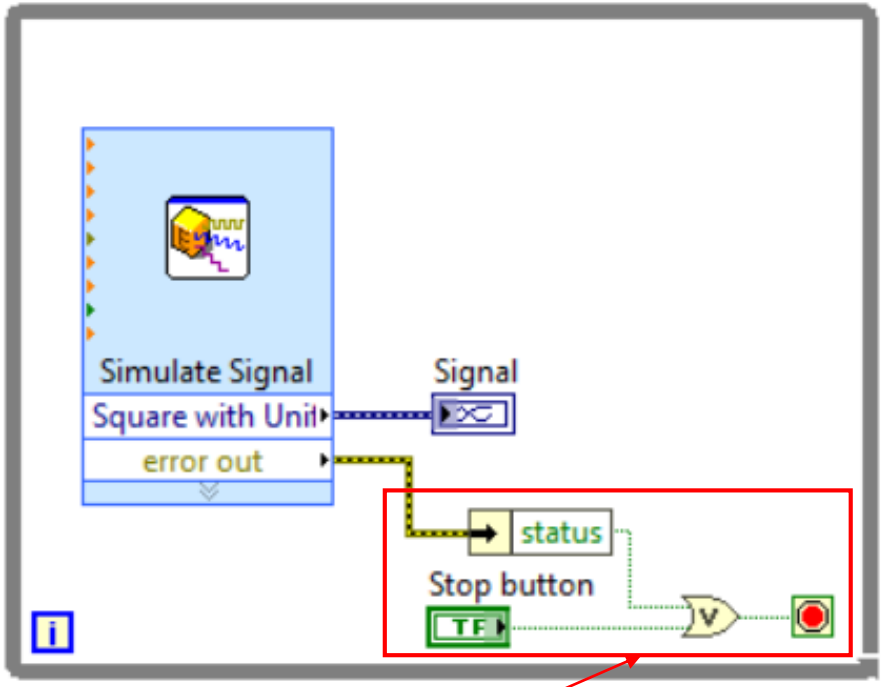
While petlja mora da se izvrši najmanje jedanput!



Itasdi

While petlja





Brojač while petlje (*iteration terminal*) – predstavlja trenutnu iteraciju petlje. Brojanje iteracija počinje od 0.

Uslovni terminali (*conditional terminal*):

- Petlja se zaustavlja kada se uslov ispuni
- Petlja se izvršava sve dok je uslov ispunjen

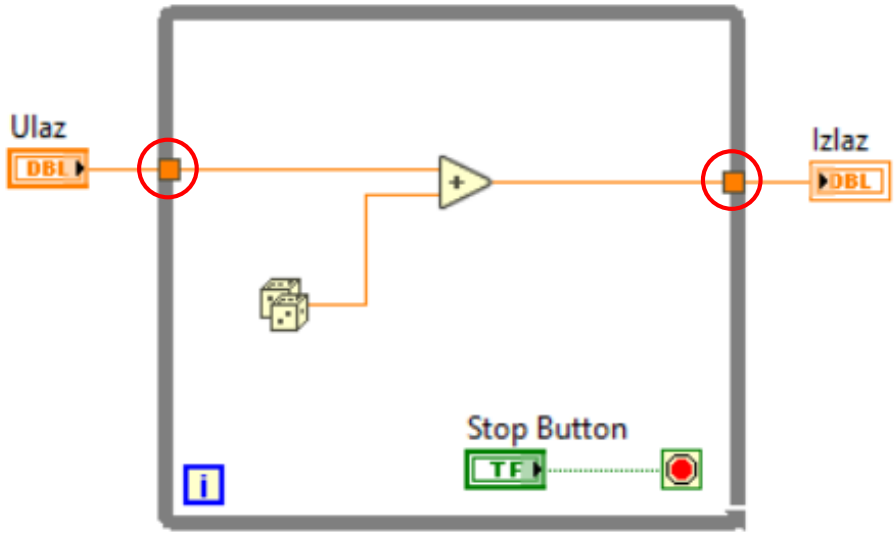
While petlja se zaustavlja ukoliko je jedan od uslova ispunjen:

- Blok za simulaciju signala prouzrokuje grešku
- Korisnik aplikacije aktivira stop dugme

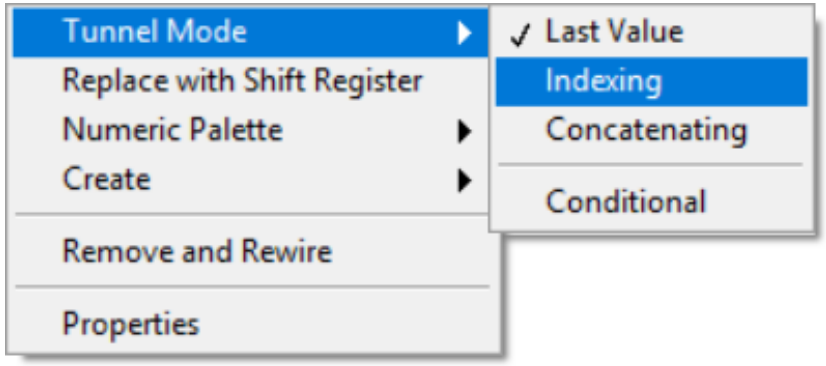
Itasdi

While petlja

Tuneli služe za dostavljanje informacija unutar i izvan while petlje.



Vrednost primenljive *Ulaz* će se pročitati samo jednom, a promenljiva *Izlaz* će se izmeniti tek nakon završetka rada while petlje.



Desni klik na tunel >> *Tunnel mode* – otvara se meni za izbor tipa tunela:

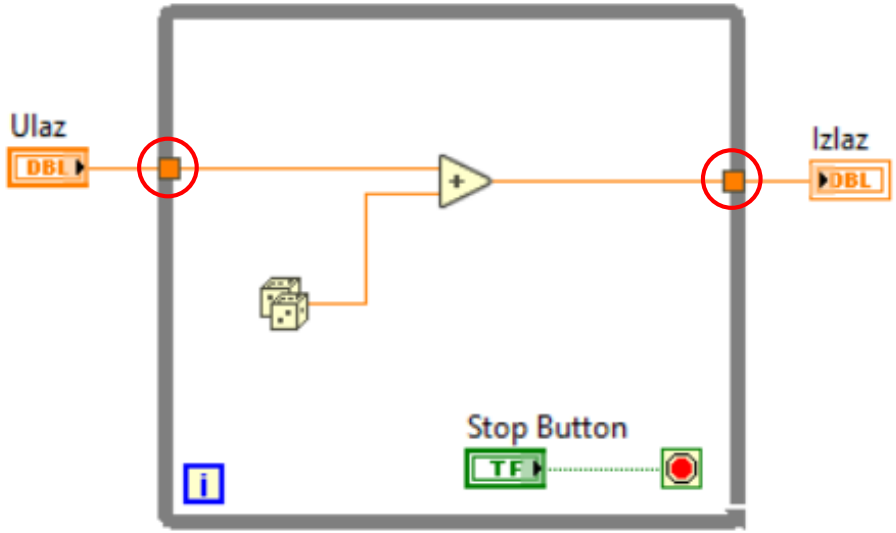
- *Last value* – čuva se samo poslednja vrednost
- *Indexing* – sve dobijene vrednosti se čuvaju kao niz
- *Concatenating* – automatski dodaje niz na kraj

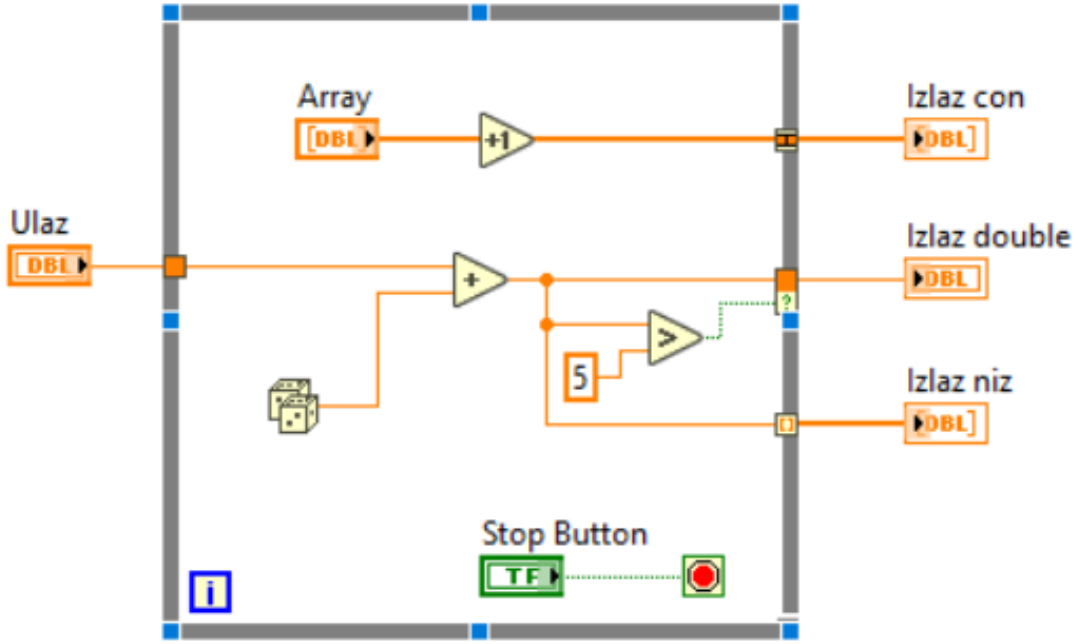
- *Conditional* – izlaz će se ispisati samo ukoliko je ispunjen uslov

Itasdi

While petlja

Tuneli služe za dostavljanje informacija unutar i izvan while petlje.



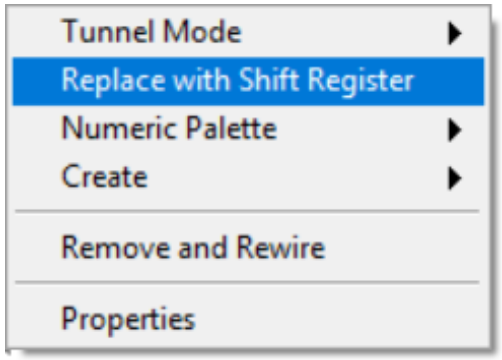


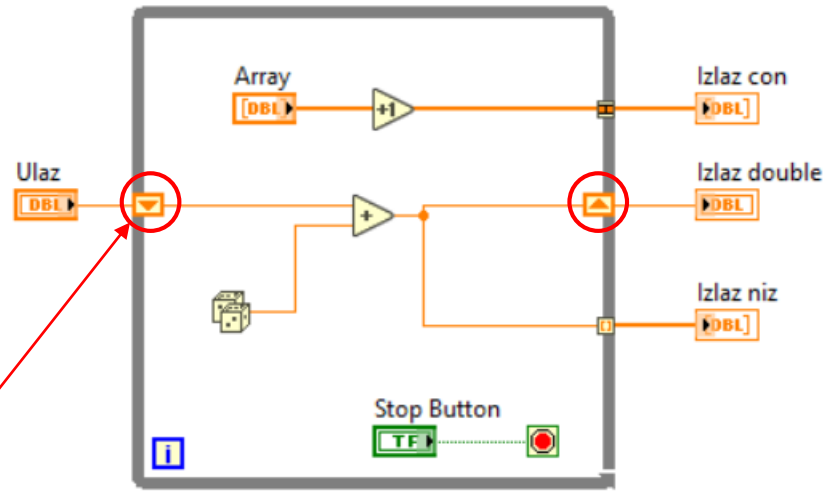
Vrednost primenljive *Ulaz* će se pročitati samo jednom, a promenljiva *Izlaz* će se izmeniti tek nakon završetka rada while petlje.

Itasdi

Shift registri

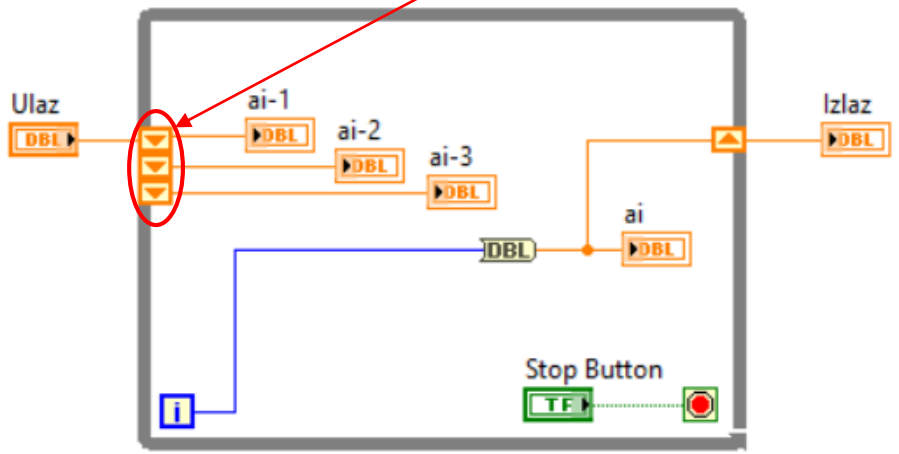
Shift registri služe da „prenesu“ informacije od izlaza ka ulazu.
Kreiranje: desni klik na tunel >> *Replace with Shift Register*.





Ukoliko se shift registar ne inicializuje, prilikom prvog pokretanja programa početna vrednost bi bila nula, ali bi se prilikom svakog sledećeg pokretanja pamtila poslednja vrednost izlaza.

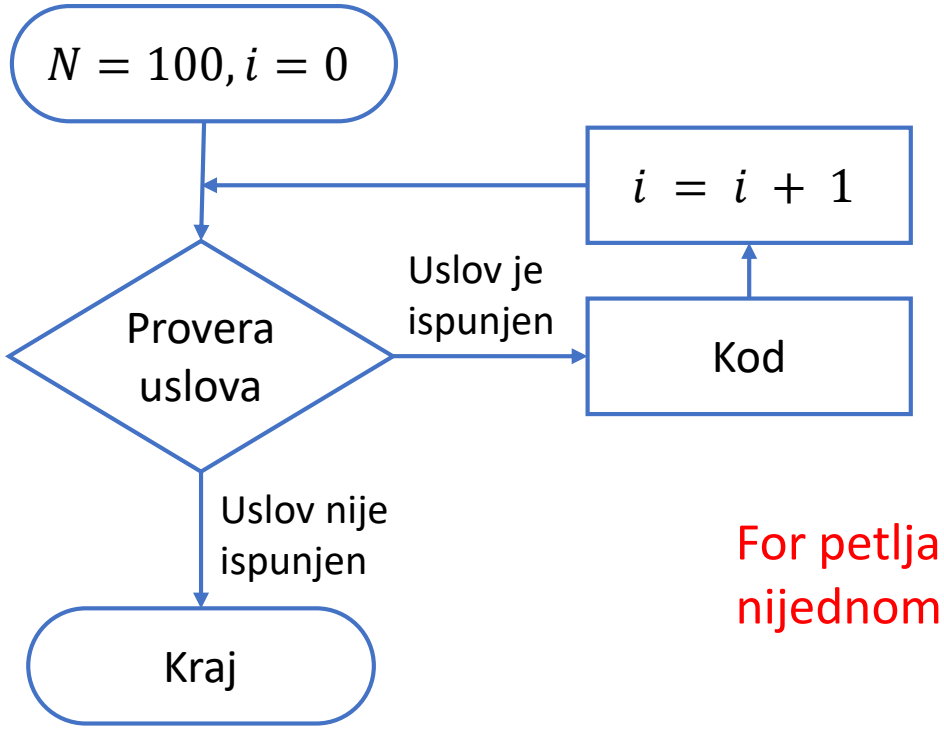
Poslednje tri vrednosti!



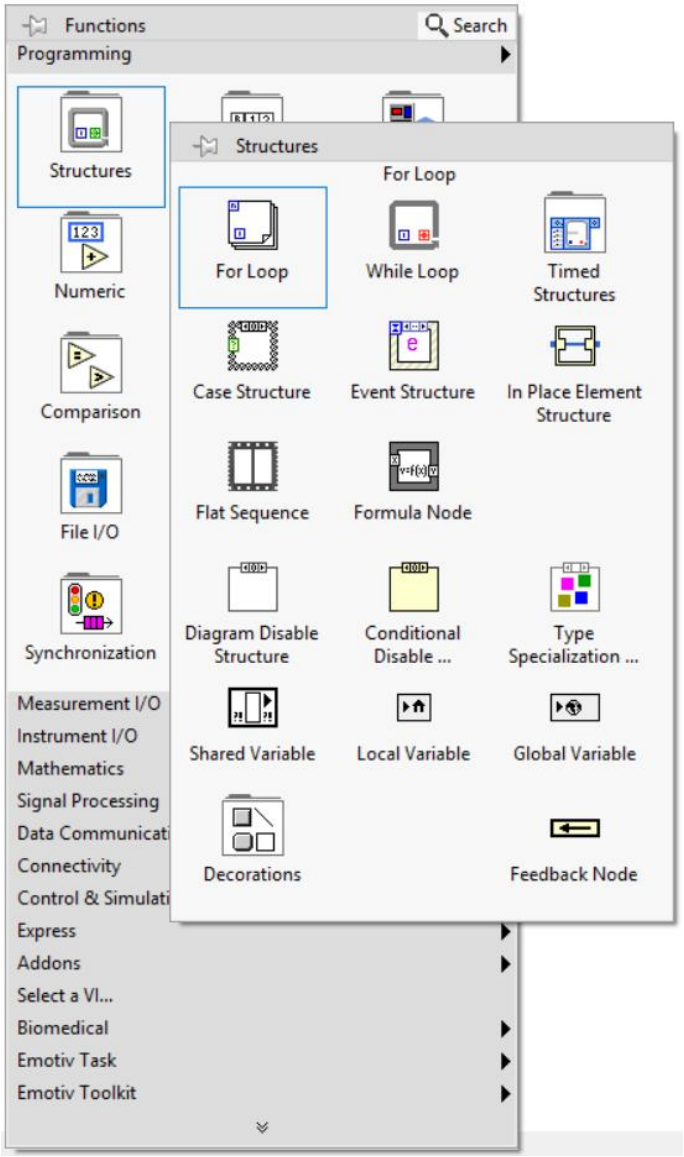
Itasdi

For petlja

Kod unutar for petlje se izvršava dok god je broj iteracija manji od nekog zadatog broja.

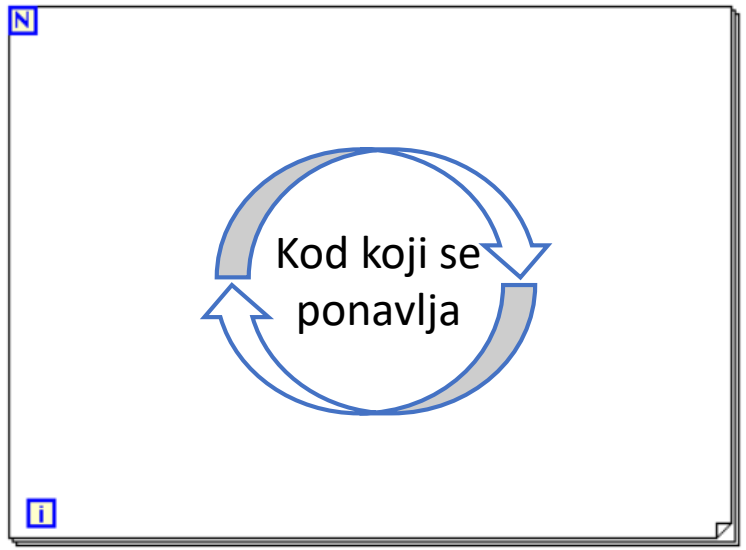


For petlja ne mora da se izvrši nijednom!

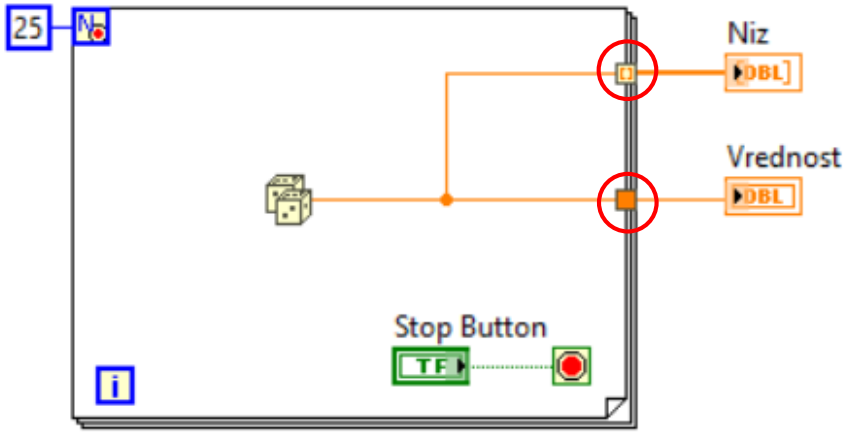


Itasdi

For petlja



Opciono se može dodati i uslovni terminal. U tom slučaju će se petlja zaustaviti ukoliko korisnik aktivira stop dugme ili ukoliko i postane veće od N.



Brojač for petlje (*iteration terminal*)

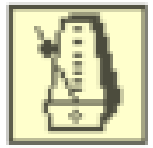
Broj iteracija (count terminal) – određuje koliko puta će se izvršiti for petlja. Ukoliko je $N \leq 0$ petlja se neće izvršavati.

Kao kod while petlje – izlaz iz for petlje može da se čuva i kao jedna vrednost i kao niz vrednosti (auto-indeksiranje).

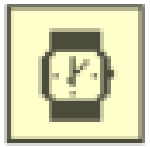
Itasdi

„Upravljanje“ vremenom

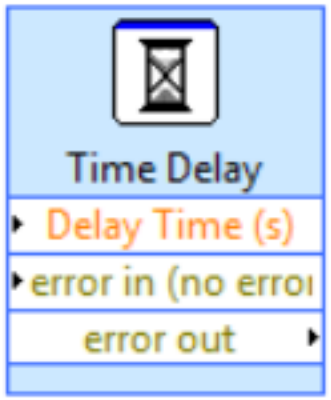
Potrebno je obezbediti pravilno trošenje resursa, tako da procesor ima vremena da radi i druge stvari.

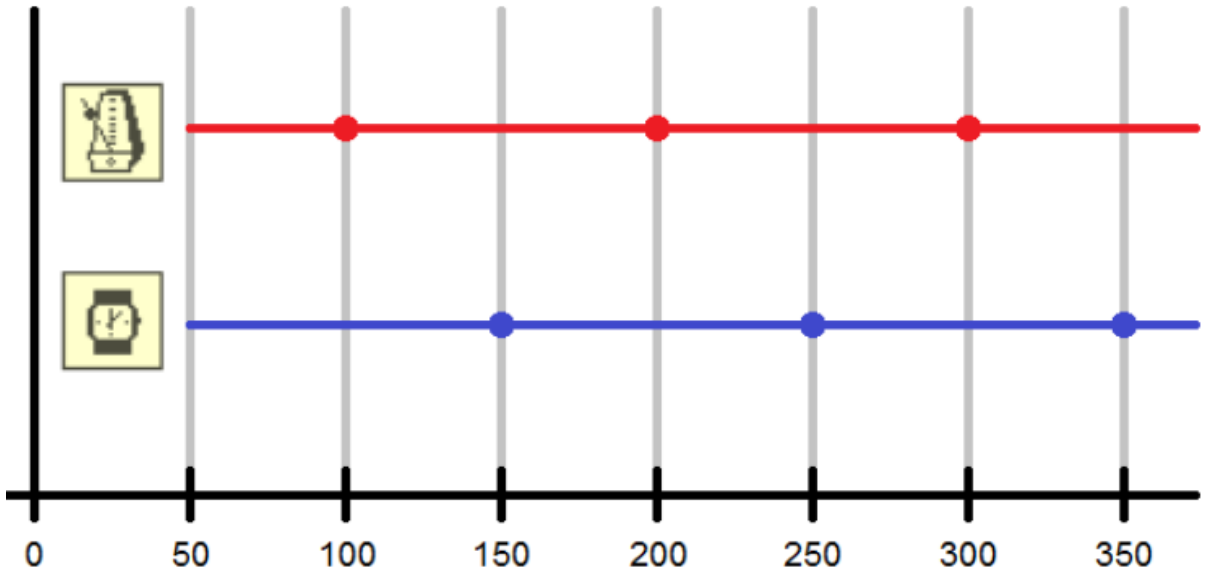


Wait Until Next ms Multiple



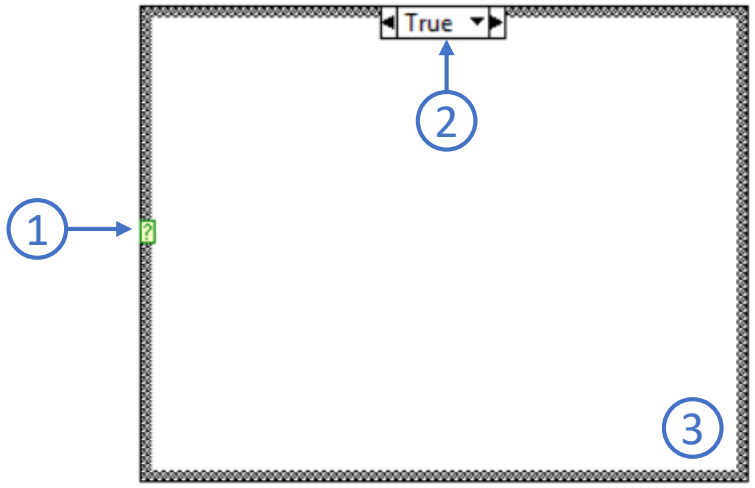
Wait (ms)



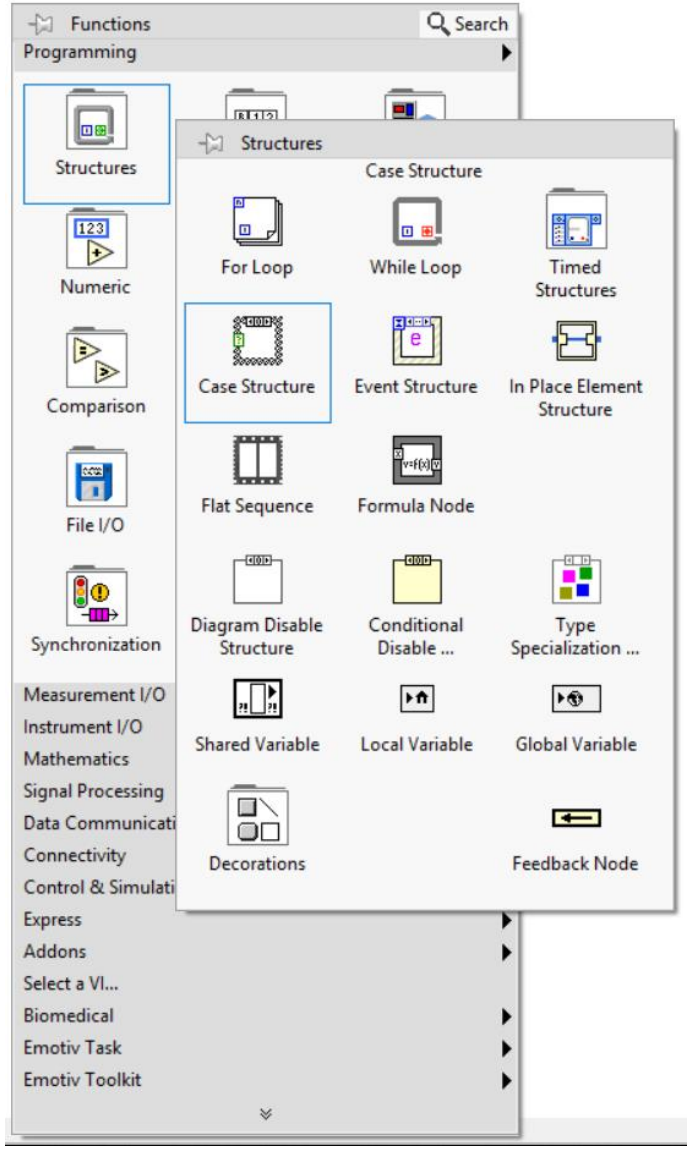


Itasdi

Case struktura

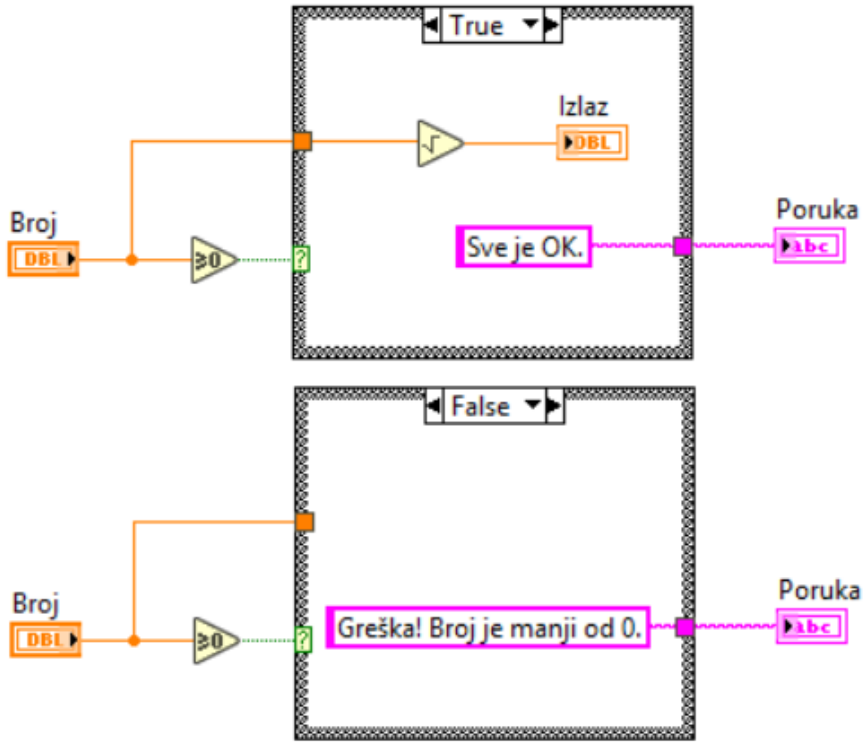


1. *Case selector* – određuje koji slučaj će se izvršiti na osnovu ulazne informacije.
2. *Selector label* – prikazuje vrednosti vezane za koje se izvršava taj slučaj. Može biti jedna vrednost ili skup vrednosti.
3. *Subdiagram* – deo koda koji se izvršava kada se vrednost povezana na *case selector* poklapa sa vrednošću u *selector label-u.*



Itasdi

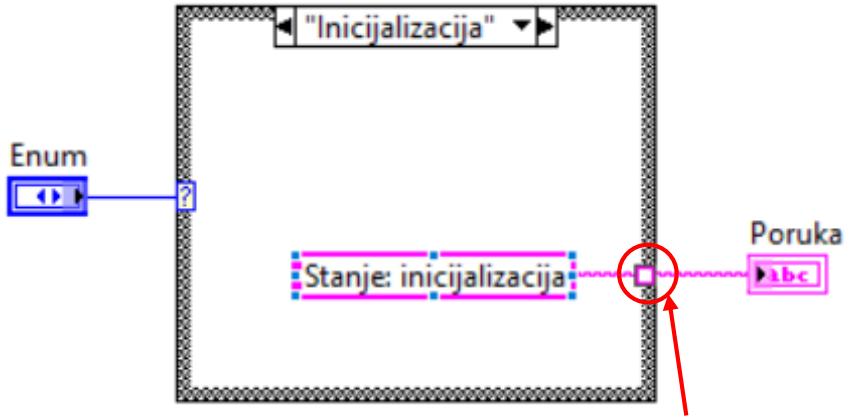
Case struktura



Vrednost *Case Selector* terminala može biti:

- Logička promenljiva (*boolean*)
- Enumerator (*Enum*)
- Numerička promenljiva (*Numeric*)
- Tekstualni podatak (*String*)
- *Error* klaster

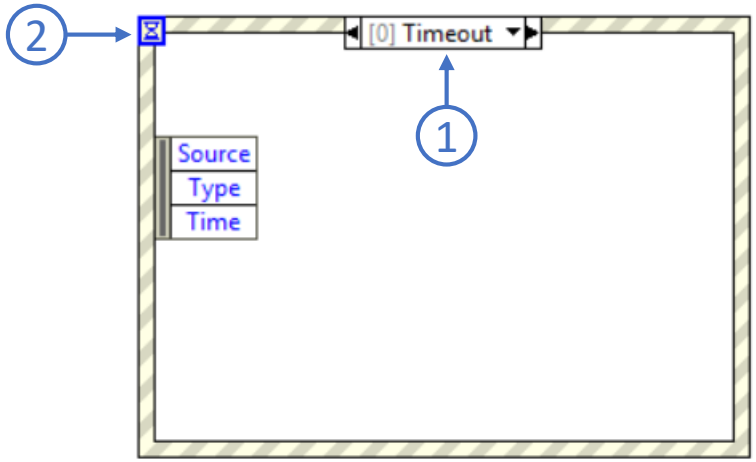
Ukoliko ne postoji slučaj za svaku vrednost *case selector* terminala mora postojati *default case*!



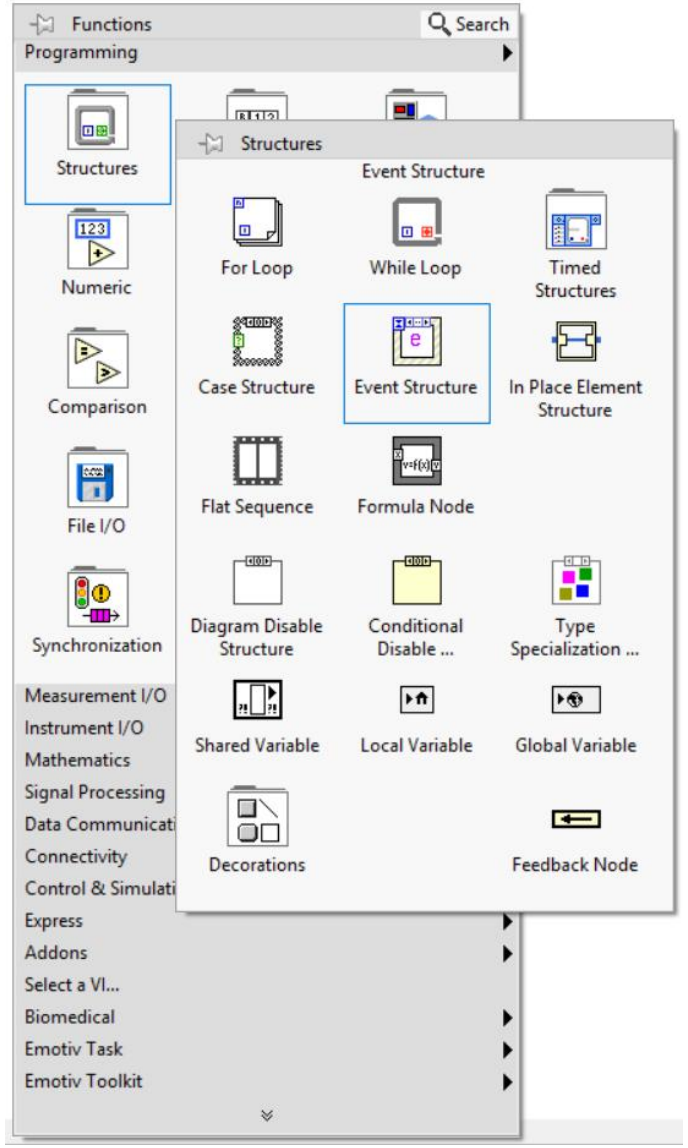
Izlazni terminali moraju imati definisane vrednosti za svaki od slučaja!

Itasdi

Event struktura



1. *Event selector label* – prikazuje koji događaj prouzrokuje izvršavanje slučaja koji je trenutno prikazan.
2. *Timeout* terminal –vreme čekanja (u ms) na događaj pre ulaska u timeout. Ukoliko se poveže vrednost na ovaj terminal, mora da postoji *Terminal event* slučaj.

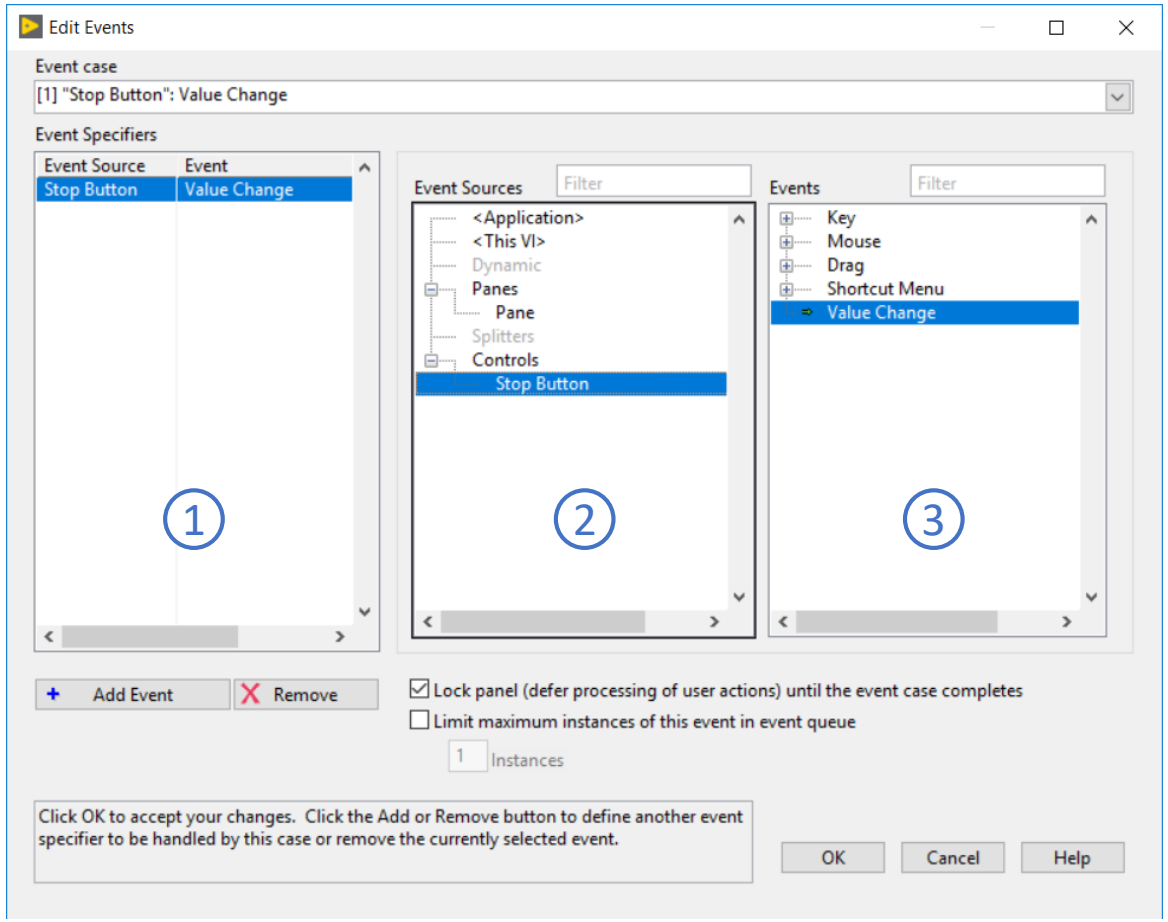


Itasdi

Event struktura

Razlika između programiranja prozivanjem (*polling*) i događajem rukovođenog programiranja (*event driven programming):*

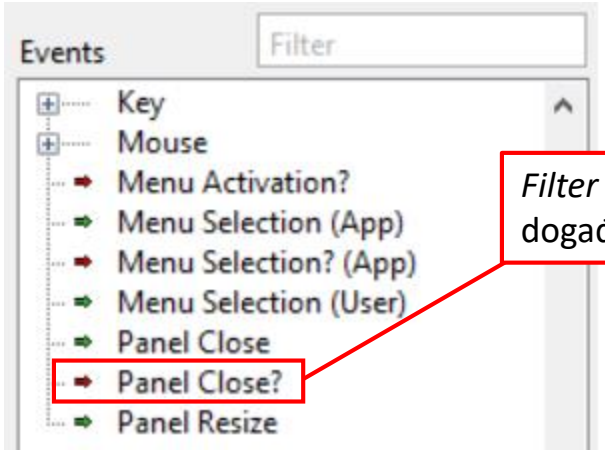
- Štedi se prostor u *Block Diagram-u*
- Program odgovara na **sve** događaje, bez gubitaka



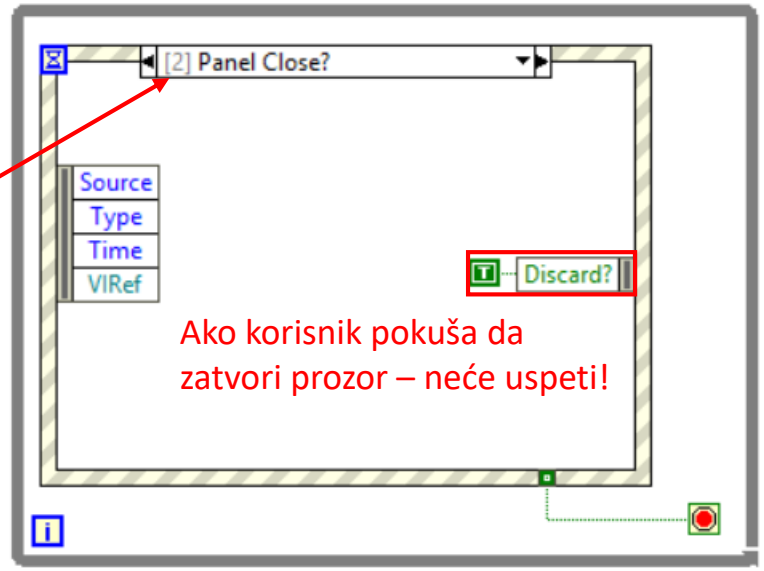
Konfigurisanje događaja ⇨ Izvor događaja ⇨ Događaj

Itasdi

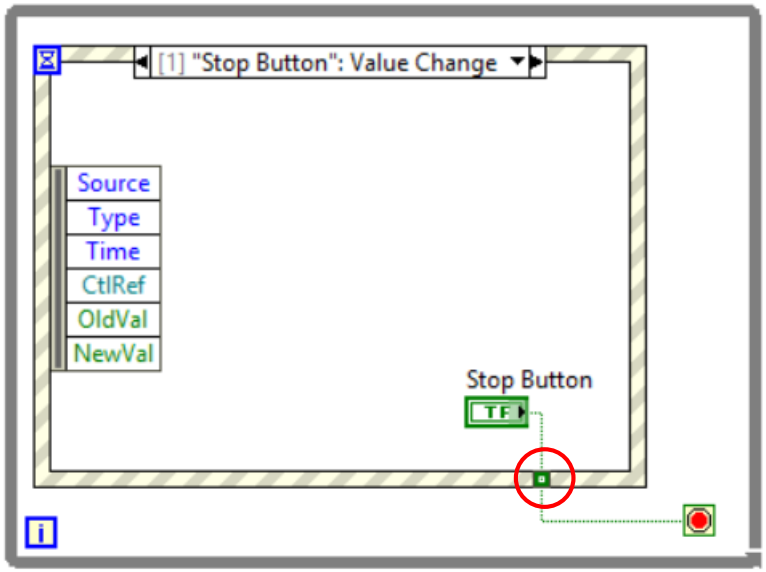
Event struktura



Filter tip događaja



Ako korisnik pokuša da zatvori prozor – neće uspeti!



Notify tip događaja – akcija korisnika se već izvršila

Filter tip događaja – korisnik je izvršio akciju, ali program odlučuje kako će da reaguje.

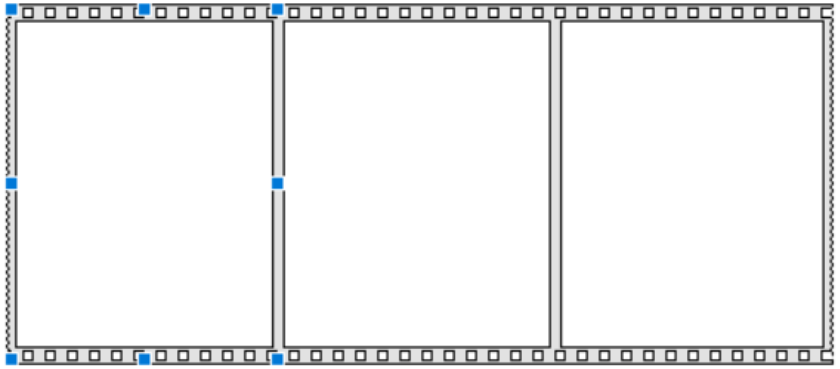
Izlazni terminali **ne** moraju imati definisane vrednosti za svaki događaj!

U jednoj while petlji staviti samo jednu event strukturu!

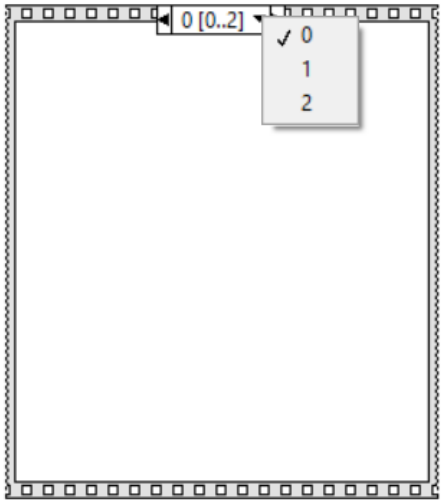
Itasdi

Sequence struktura

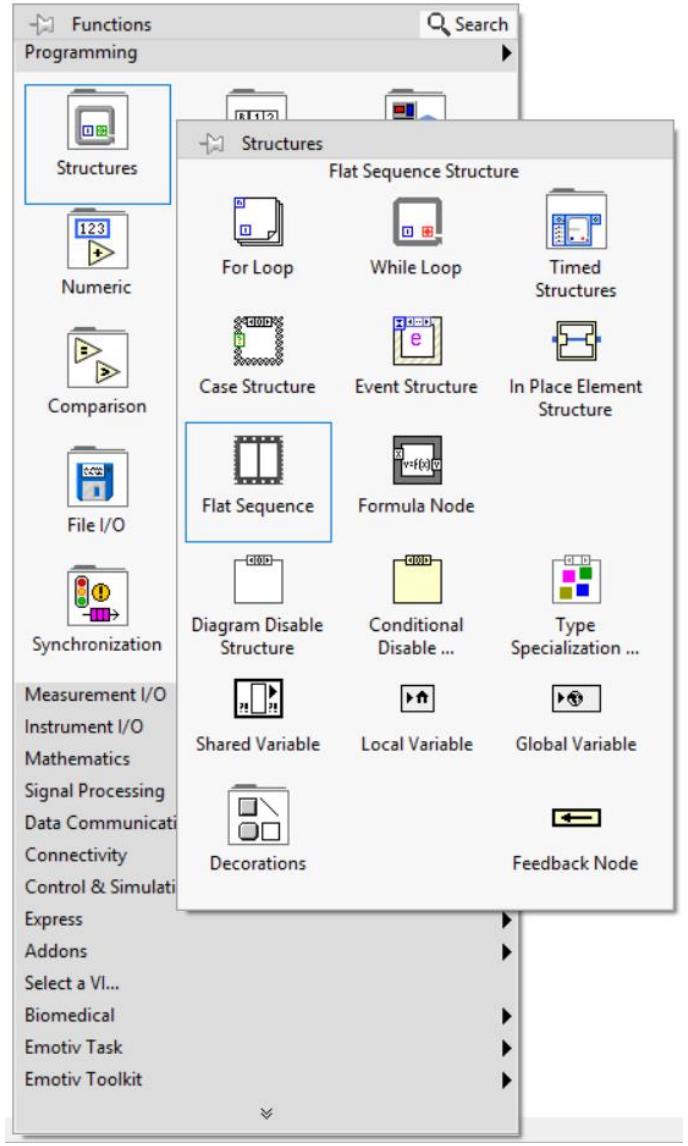
Sequence struktura se sastoji od više poddijagrama koji se izvršavaju sekvencijalno. Korišćenjem ove strukture će biti osigurano da će se akcije izvršavati jedna za drugom.



Flat sequence



Stacked sequence



Hvala na pažnji!

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