Itasdi

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

Praktikum iz merno-akvizicionih sistema

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

Itasdi

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

Lekcija 3

Strukture podataka

Cilj

Cilj lekcije je da studenti savladaju:

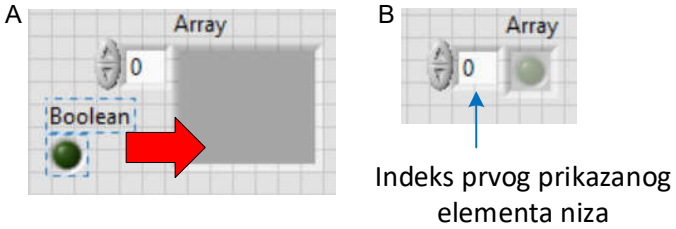
- manupulaciju nizovima
- pojam dinamičkog tipa podataka
- kreiranje lokalnih promenljivih
- programsko kontrolisanje osobina kontrola/indikatora
- mogućnosti selekcije sekvenci *Case* strukture različitim tipovima podataka
- manupulaciju klasterima
- striktno definisanje izgleda kontrola i indikatora.

3.1 Nizovi

Zadatak 3.1.1.
Kreirati program koji konvertuje vrednosti logičkog niza u numerički niz, tj. koji TRUE/FALSE vrednosti konvertuje u 0/1 vrednosti.

Uputstvo:

1. Otvoriti nov **.vi** program.
2. U paleti ***Controls»Modern»Boolean*** izabrati logičku kontrolu (*Round LED*) i postaviti je na *Front Panel*.
3. U paleti ***Controls»Modern»Array, Matrix & Cluster*** izabrati *Array* i postaviti na *Front Panel*.
4. Pomoću pomoću *Position/Size/Select* alatke  selektovati *Round LED* kontrolu i držeći pritisnut levi klik miša „prevući“ je u *Array* kao što je prikazano na Sl. 3.1.1A. Rezultat „prevlačenja“ je prikazan na Sl.3.1.1B.

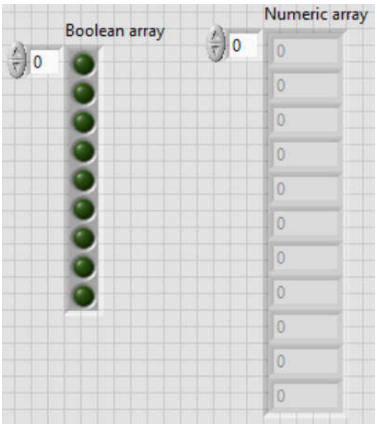


Sl. 3.1.1. A) Postupak kreiranja niza logičkih kontrola, B) Kreiran logički niz kontrola

VAŽNA NAPOMENA: Broj dimenzija niza se može uvećati tako što se klikne desnim tasterom miša na indeks (Sl. 3.1.1) i izabere opcija *Add Dimension*. Broj dimenzija se redukuje izborom opcije *Remove Dimension* (minimalna broj dimenzija je 1).

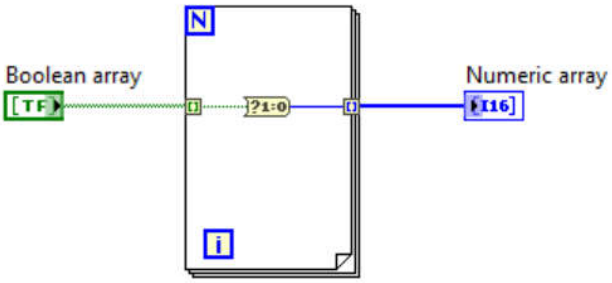
5. Postupak kreiranja niza ponoviti za niz numeričkih indikatora (*Numeric Indicator* u paleti ***Controls»Modern»Numeric***).
6. Razvući niz logičkih kontrola i niz numeričkih indikatora tako da se vidi više članova niza, Sl. 3.1.2. Pomoću alatke *Labeling Tool* u *Tools* paleti uneti odgovarajuće natpise za labele niza kontrola i indikatora.

   VAŽNA NAPOMENA: Nije moguće napraviti niz nizova.



Sl. 3.1.2. Niz logičkih kontrola i niz numeričkih indikatora

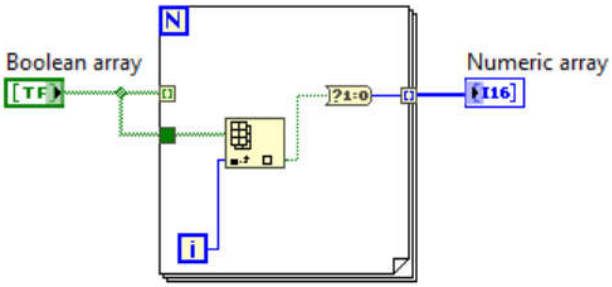
7. U *Block Diagram*-u kreirati programski kôd koji primenom *For* petlje selektuje elemente logičkog niza i konvertuje ih pomoću funkcije ***Functions»Programming»Boolean» Boolean To (0,1)*** u broj 0 ili 1. Programski kôd je prikazan na Sl. 3.1.3. Povezivanje *Boolean array* kontrole sa *For* petljom ostvariti pomoću alatke *Wire Tool*. Primetiti pojavu „uglastih zagrada“ na ivici *For* petlje. Ova pojava predstavlja tzv. auto-indeksiranje, tj. *For* petlja izdvaja element po element niza koji dalje može da se koristi za obradu unutar kôda *For* petlje. Takođe, i na „izlazu“ *For* petlje može se primetiti ista pojava auto-indeksiranja pri povezivanju sa *Numeric array* indikatorom, tj. obrađeni elementi (u ovom slučaju elementi nastali nakon konverzije TRUE/FALSE u 0/1) se pakuju u izlazni niz. Uočiti da brojač *For* petlje nije definisan eksplicitno već da je definisan implicitno zahvaljujući auto-indeksiranju.



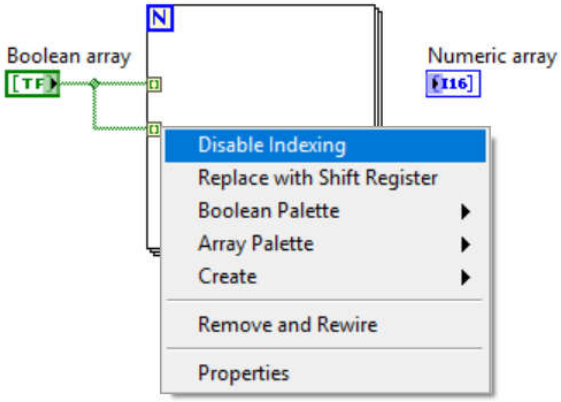
Sl. 3.1.3. Auto-indeksiranje u *For* petlji

8. Sačuvati program kao *03_01_Konverzija_logickog_niza_u_numericki.vi*.
9. Na *Boolean array* kontroli upaliti nekoliko lampica, a potom pokrenuti izvršavanje programa. Uočiti da će se odgovarajuće vrednosti „1“ pojaviti i na *Numeric array* indikatoru. Indeks prvog prikazanog elementa niza (Sl. 3.1.1B) se može menjati, tj. elementi se mogu „listati“ - uočiti kako se pri njegovoj promeni pomeraju i elementi niza.

10. Programski kôd sa Sl. 3.1.3 je ekvivalentan kôdu na Sl. 3.1.4 gde je umesto izlaza auto-indeksiranja iskorišćen izlaz funkcije ***Functions»Programming»Array »Index Array*** za dalju konverziju. Auto-indeksiranje se isključuje tako što se uradi desni klik miša na „uglaste zagrade“ i izabere opcija *Disable Indexing,* Sl. 3.1.5.

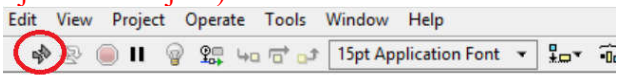


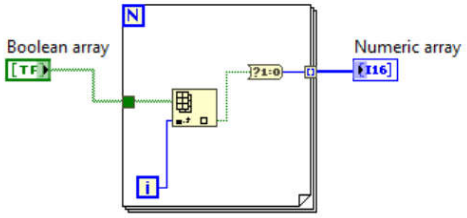
Sl. 3.1.4. Primena *Index Array* funkcije



Sl. 3.1.5. Isključivanje auto-indeksiranja

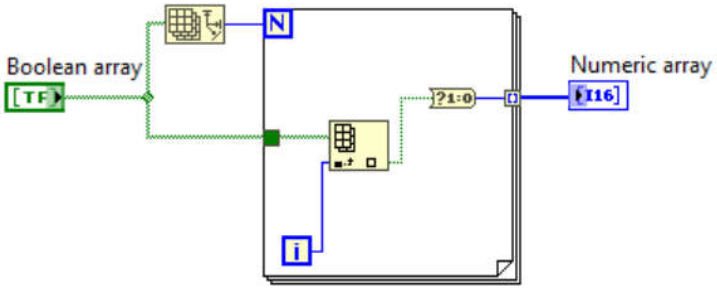
11. Kreirati programski kôd sa Sl. 3.1.4. Uočiti da su od *Boolean array*-a na *For* petlju priključene obe žica: i sa i bez auto-indeksiranja. Pokrenuti izvršavanje programskog kôd-a.
12. Zaustaviti izvršavanje programa. Izbrisati žicu za auto-indeksiranje kao na Sl. 3.1.6. Uočiti da će se pojaviti slomljena strelica u *Status Toolbar*-u. Zašto? (Pomoć: na šta je vezan brojač?)





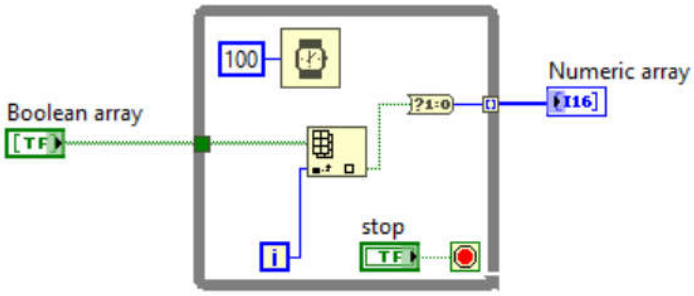
Sl. 3.1.6.

13. Kreirati programski kôd kao na Sl. 3.1.7 koji koristi funkciju ***Functions»Programming»Array»Array Size*** za određivanje dimenzije niza. Pokrenuti program.



Sl. 3.1.7. Primena *Array Size* funkcije

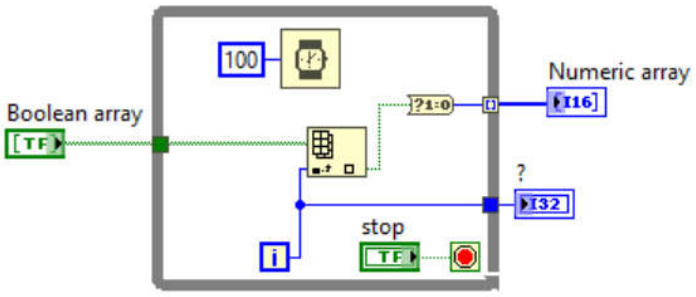
14. Zaustaviti program. Zameniti *For* petlju *While* petljom, Sl. 3.1.8. Nasumično uključiti lampice i pokrenuti program. Uočiti da će tek nakon pritiska na dugme STOP da se pojave prave vrednosti na *Numeric array*.



Sl. 3.1.8. Auto-indeksiranje u *While* petlji

15. Zaustaviti program. Modifikovati ga kao na Sl. 3.1.9. Prenos podataka kroz petlju, bilo u nju ili van nje, naziva se „tunelovanje“. Pokrenuti program i zaustaviti ga. Šta će da pokazuje indikator označen sa „?“?

VAŽNA NAPOMENA: Mogućnosti auto-indeksiranja i „tunelovanja“ (tj. prenosa podataka u petlju ili iz petlje) važe podjednako i za *For* petlju i za *While* petlju.

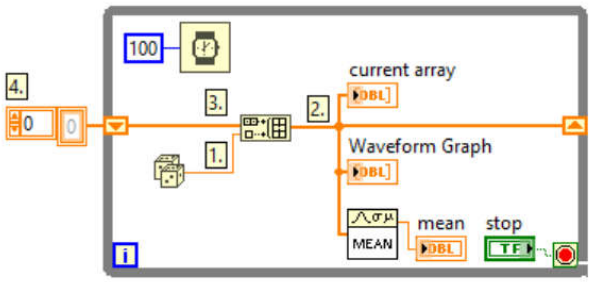


Sl. 3.1.9. „Tunelovanje“ tj. prenos podataka kroz petlju

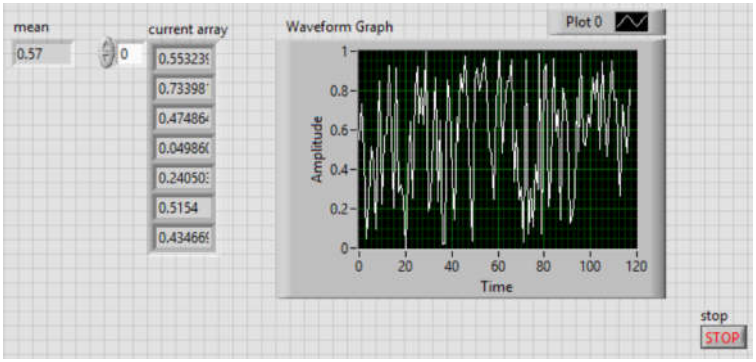
16. Sačuvati program kao *03_01_Konverzija_logickog_niza_u_numericki_While.vi*.
17. Proučiti pomoću *Context Help*-a šta rade i druge funkcije za manipulaciju nad nizovima a koje se nalaze u paleti ***Functions» Programming»Array***.

Zadatak 3.1.2.
Kreirati program koji koristeći *Shift* registar u *While* petlji dinamički gradi niz slučajnih podataka. U toku izvršavanja programa prikazivati srednju vrednost niza na numeričkom indikatoru, a niz prikazivati na indikatoru ***Controls»Modern»Waveform Graph***. Obezbediti da procesor ima slobodno vreme (trajanje jedne iteracije *While* petlje je podešeno na 100 ms).

Uputstvo:
1. Otvoriti nov **.vi** program.
2. Kreirati *While* petlju koja se zaustavlja pritiskom na dugme STOP. Funkcijom *Wait (ms)* obezbediti da procesor ima slobodno vreme i da svaka iteracija traje 100 ms (Lekcija 2.3).
3. Iz palete ***Functions»Mathematics*** izabrati funkciju *Random Number 0-1* za generisanje slučajnih brojeva i uneti je u *While* petlju.
4. Funkcija za gradjenje niza je ***Build Array*** u paleti ***Functions» Programming»Array***. Razvlačenjem proširiti ovu funkciju tako da ima dva ulaza. Dodati *Shift* registar na *While* petlju na način objašnjen u Lekciji 2.5.2. *Block diagram* celog programa je prikazan na Sl. 3.1.10. Brojevima 1, 2, 3 i 4 je označen redosled povezivanja žica za *Build Array* funkciju i *Shift* registar. Uočiti da je najpre potrebno da se poveže izvor slučajnih podatka na donji priključak *Build Array* funkcije (1.), potom izlaz *Build Array* funkcije na desnu stranu *Shift* registra (tj. „napuniti“ *Shift* registar tipom podatka) (2.), a potom levu stranu *Shift* registra vratiti na gornji ulaz *Build* funkcije (3.). Inicijalizacija vrednosti *Shift* registra se postiže tako što se miš pozicionira iznad levog *Shift* registra, uradi desni klik i izabere opcija *Create Constant* (4.).
5. *Front Panel* programa je prikazan na Sl. 3.1.11. Za određivanje srednje vrednosti niza izabrati funkciju ***Mean*** u paleti ***Functions»Mathematics»Probability &Statistics***.



Sl. 3.1.10. Dinamičko građenje niza. Redosled povezivanja priključaka *Build Array* funkcije na *Shift* registar je dat brojevima 1., 2., 3., 4.



Sl. 3.1.11. *Front Panel* programa za dinamičko građenje niza

6. Sačuvati program kao *03_01_Dinamicko_gradjenje_niza*.
7. Pokrenuti izvršavanje programa. Kolika je srednja vrednost niza koji se gradi?
8. Zaustaviti progam i zatvoriti ga.

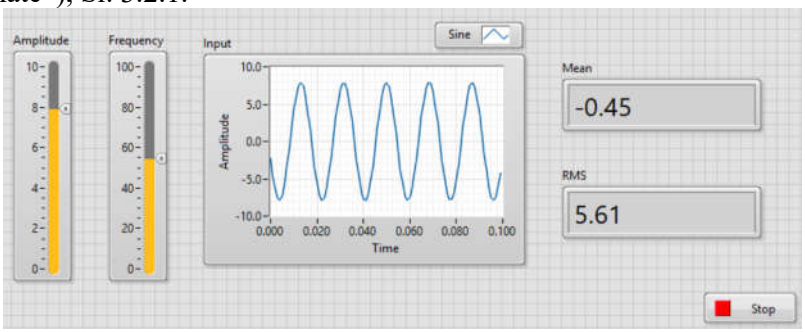
Zadatak za samostalni rad 3.1.3.
Kreirati program koji:
1. sabira niz [1,2,3,4] sa brojem 4.
2. sabira niz [1,2,3,4] sa nizom [4,3,2,1]
3. sabira niz [1,2,3,4] sa nizom [4,3,2].

Diskutovati rezultate sabiranja.

VAŽNA NAPOMENA: LabVIEW funkcije imaju osobinu **polimorfizma**, tj. omogućavaju da se na njihove ulaze dovode različiti tipovi promenljivih. Na primer, funkcija sabiranja niza sa brojem i niza sa nizom je identična po svom izgledu.

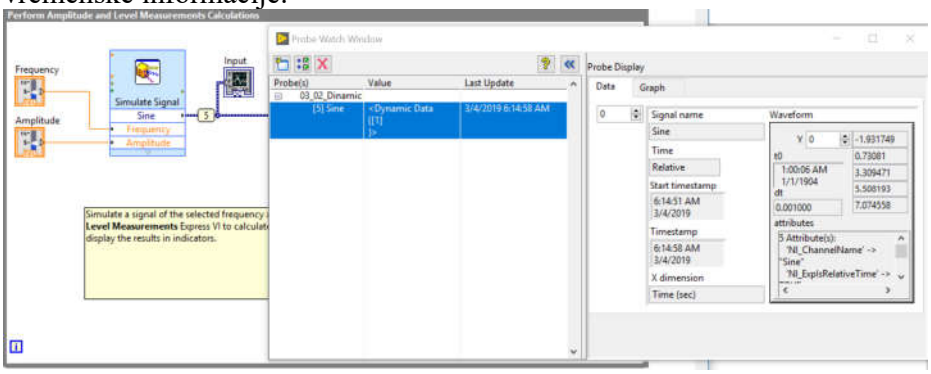
3.2 Dinamički tip podataka

1. Otvoriti primer *Express VI - Amplitude and Level Measurements.vi* koji se nalazi u ***Help»Find Examples*** (radi lakšeg nalaženja ovog primera u *Search* sekciji uneti „simulate“), Sl. 3.2.1.



Sl. 3.2.1. *Front Panel* primera *Express VI - Amplitude and Level Measurements.vi*

2. Proučiti *Block Diagram* primer-a. Pokrenuti program i menjati amplitudu i frekvenciju signala prateći promene na ekranu. Pomoću alatke *Probe* posmatrati sadržaj izlaza iz funkcije *Simulate Signal*, Sl. 3.2.2. Ova *Express* funkcija daje tzv. dinamički tip podataka koji osim odbiraka signala ima i vremenske informacije.

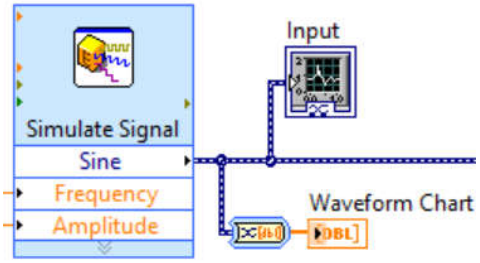


Sl. 3.2.2. Struktura dinamičkog tipa podataka

3. Zaustaviti program. Sačuvati ga pomoću *Save As* opcije kao *03_02_Dinamicki_tip_podataka.vi*.

Zadatak 3.2.1.
Polazeći od programa *03_02_Dinamicki_tip_podataka.vi*. kreirati program koji konvertuje dinamički tip podataka u numerički niz (koristeći funkciju ***Functions»Express»Convert from Dynamic Data***) i prikazuje na *Waveform Chart*-u.

Uputstvo:
1. Otvoriti fajl *03_02_Dinamicki_tip_podataka.vi* sačuvan u prethodnoj tački.
2. Modifikovati program kao što je prikazano na Sl. 3.2.3.



Sl. 3.2.3. Konverzija dinamičkog tipa podataka u numerički niz

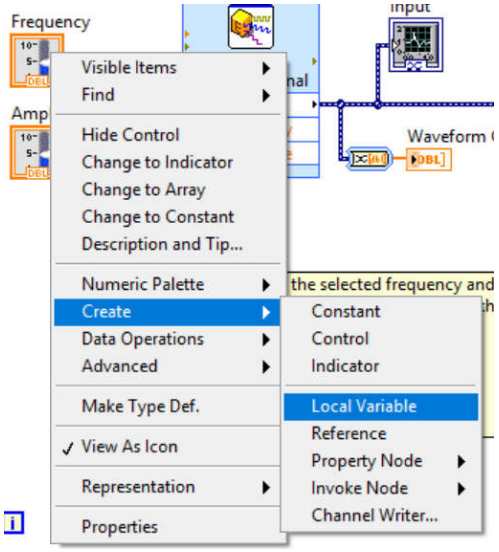
3. Pokrenuti program. Po čemu se razlikuje prikaz na grafičkom indikatoru *Input* u odnosu na prikaz na kreiranom *Waveform Chart*-u?
4. Zaustaviti program i **sačuvati izmene**.

3.3 Lokalne promenljive – primena u inicijalizaciji

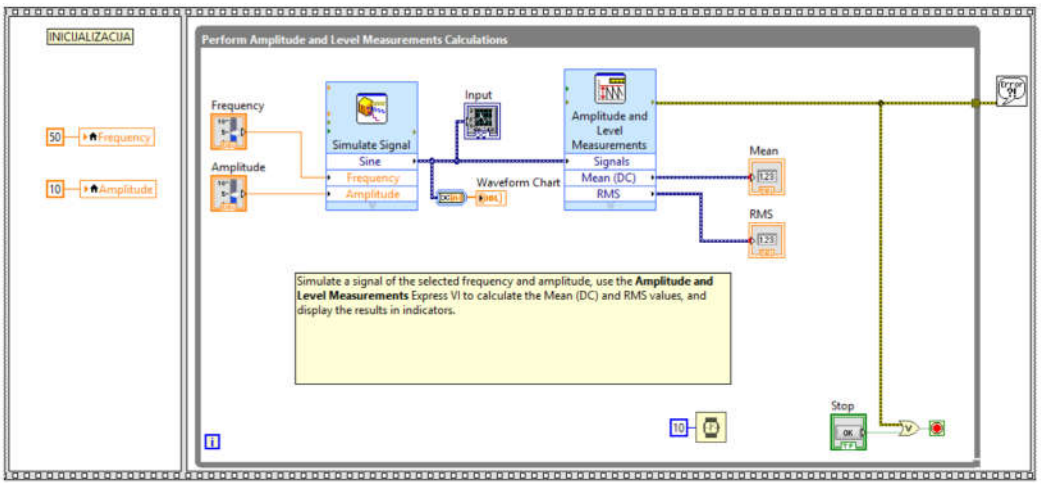
Zadatak 3.3.1.
Polazeći od sačuvanog programa *03_02_Dinamicki_tip_podataka.vi*. kreirati program koji na početku izvršavanja radi INICIJALIZACIJU koja podrazumeva zadavanje početnih vrednosti kontrola *Amplitude* i *Frequency*, a potom nastavlja sa regularnim radom.

Uputstvo:
1. Otvoriti fajl *03_02_Dinamicki_tip_podataka.vi* sačuvan u prethodnoj tački.
2. Ceo program postaviti u *Sequence* strukturu (Lekcija 2.2). Potom dodati jedan frejm *Sequence* strukture pre tog i u njemu napraviti lokalne promenljive kontrola *Amplitude* i *Frequency*. Lokalne promenljive se kreiraju tako što se miš pozicionira iznad kontrole, potom se klikne desnim mišem i izabere opcija ***Create»Local Variable***, Sl. 3.3.1. Kreirane lokalne promenljive pomoću *Position/Size/Select* alatke  selektovati i postaviti u nultu sekvencu *Sequence* structure kao na Sl.3.3.2.
3. Pokrenuti program. Uočiti da je početna vrednost kontrola sada kao što je zadato lokalnim promenljivama.
4. Zaustaviti program i **sačuvati izmene**.

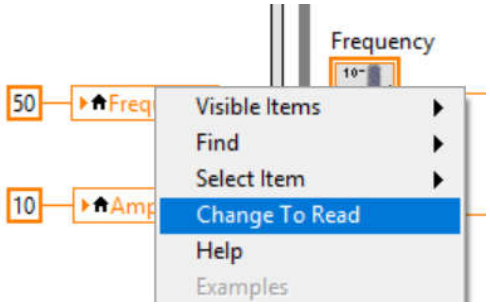
VAŽNA NAPOMENA: Lokalne promenljive se na identičan način prave i za kontrole i za indikatore. U njih se može upisivati, ali i čitati. Lokalna promenljiva u koju se upisuje se može pretvoriti u promenljivu iz koje se čita tako što se miš pozicionira iznad nje, klikne se desnim mišem i izabere opcija ***Change to Read***, Sl. 3.3.3. Slično je i kada treba promeniti namenu na ***Change to Write.***



Sl. 3.3.1. Kreiranje lokalne promenljive



Sl. 3.3.2. Primena lokalnih promenljivih za inicijalizaciju



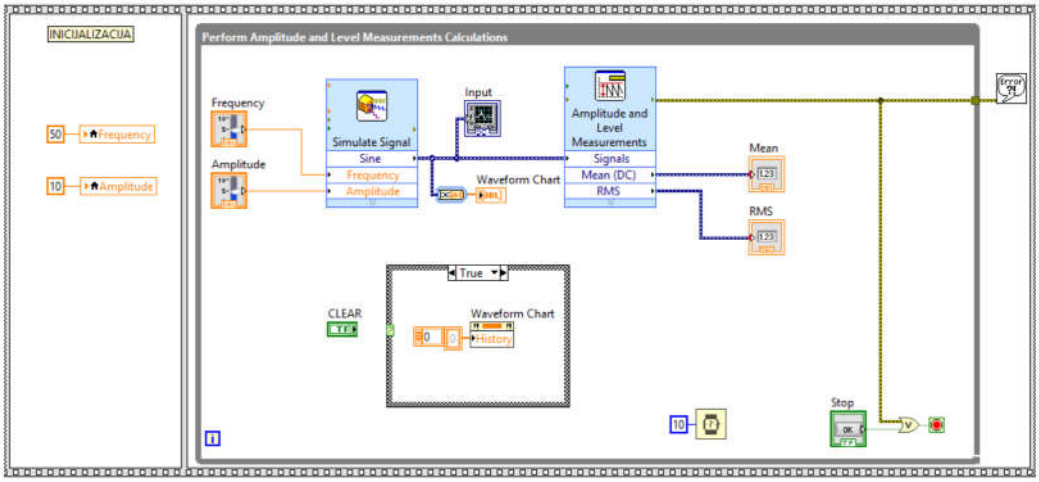
Sl. 3.3.3. Promena lokalne promenljive iz môda upisa u môd čitanja

3.4 Programsko kontrolisanje osobina kontrola/indikatora

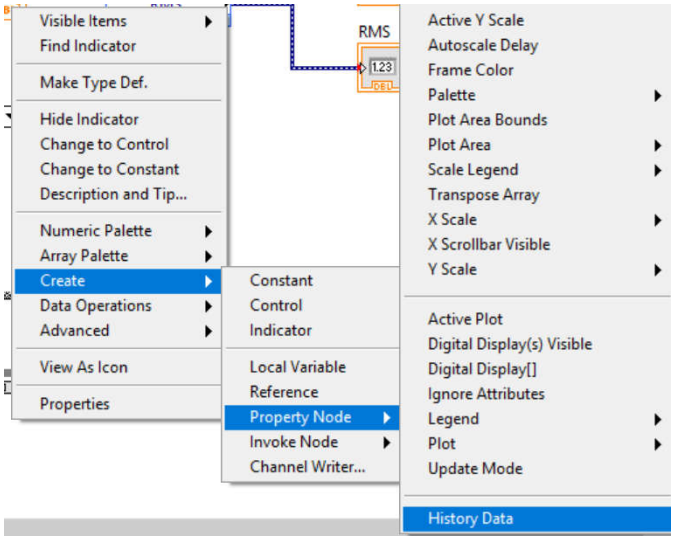
Zadatak 3.4.1.
Polazeći od sačuvanog programa *03_02_Dinamicki_tip_podataka.vi*. kreirati program koji pri pritisku na dugme CLEAR briše sadržaj *Waveform Chart*-a.

Uputstvo:

1. Otvoriti fajl *03_02_Dinamicki_tip_podataka.vi* sačuvan u prethodnoj tački.
2. Unutar *While* petlje kreirati *Case* strukturu kao i dugme CLEAR koje treba povezati na *Case selector* ulaz *Case* strukture, Sl. 3.4.1. Unutar TRUE sekvence *Case* strukture kreirati ***Property Node»History Data*** indikatora *Waveform Chart* tako što se miš pozicionira iznad indikatora, potom se klikne desnim mišem i izabere opcija ***Create» Property Node»History Data***, Sl. 3.4.2.



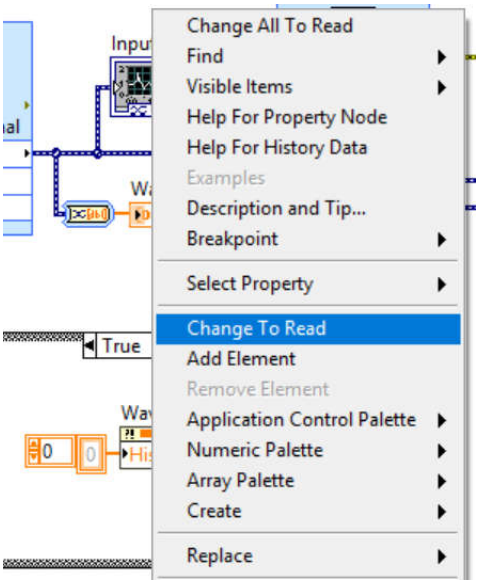
Sl. 3.4.1. Programsko podešavanje osobine kontrole/indikatora



Sl. 3.4.2. Kreiranje *Propery Node*-a kontrole/indikatora

3. Pokrenuti program. Uočiti da se klikom na dugme CLEAR briše sadržaj *Waveform Chart*-a.
4. Zaustaviti program i **sačuvati izmene**.

VAŽNA NAPOMENA: *Property Node* se na identičan način prave i za kontrole i za indikatore. U njih se može upisivati, ali i čitati. *Property Node* u koji se upisuje se može pretvoriti u promenljivu iz koje se čita tako što se miš pozicionira iznad nje, klikne se desnim mišem i izabere opcija ***Change to Read***, Sl. 3.4.3. Slično je i kada treba promeniti namenu na ***Change to Write.***



Sl. 3.4.3. Promena Property Node-a iz môda upisa u môd čitanja

Zadatak za samostalan rad 3.4.2.

Modifikovati Zadatak 3.4.1. tako se umesto pritiskom na dugme CLEAR, sadržaj *Waveform Chart*-a menja:

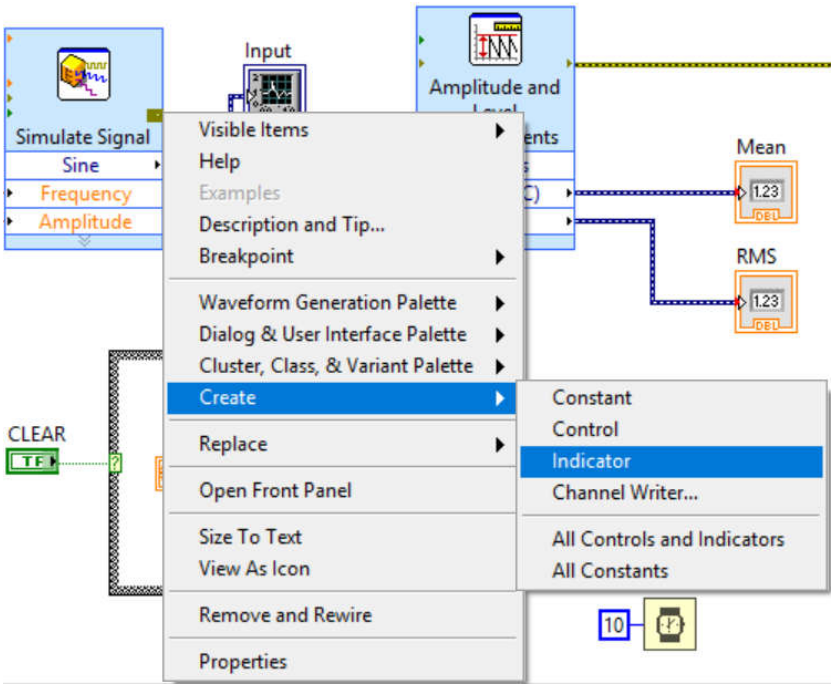
1) selekcijom odgovarajuće vrednosti *String* kontrole
2) selekcijom odgovarajuće vrednosti *Enum* kontrole.

3.5 Klasteri

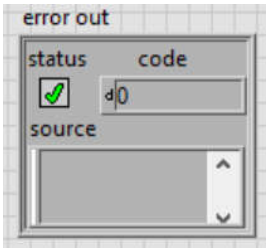
Klasteri predstavljaju strukturu podataka koja se sastoji iz više različitih tipova podataka.

Zadatak 3.5.1.
Modifikovati program iz prethodne tačke tako da se završava ili pritiskom na dugme STOP ili u slučaju da *status* greške postane TRUE.

1. Otvoriti fajl *03_02_Dinamicki_tip_podataka.vi* sačuvan u prethodnoj tački.
2. Pozicionirati alatku *Wire Tool* iznad „*error out*“ izlaza *Express* funkcije *Simulate Signal*. Kliknuti desnim mišem i izabrati opciju ***Create»Indicator***, Sl. 3.5.1.
3. *Error* klaster se sastoji iz tri podatka: *status* (logička promenljiva koja pokazuje da li je došlo do greške), *code* (numerička promenljiva I32 koja je kôd greške) i *source* (*string* promenljiva koja opisuje izvor greške).

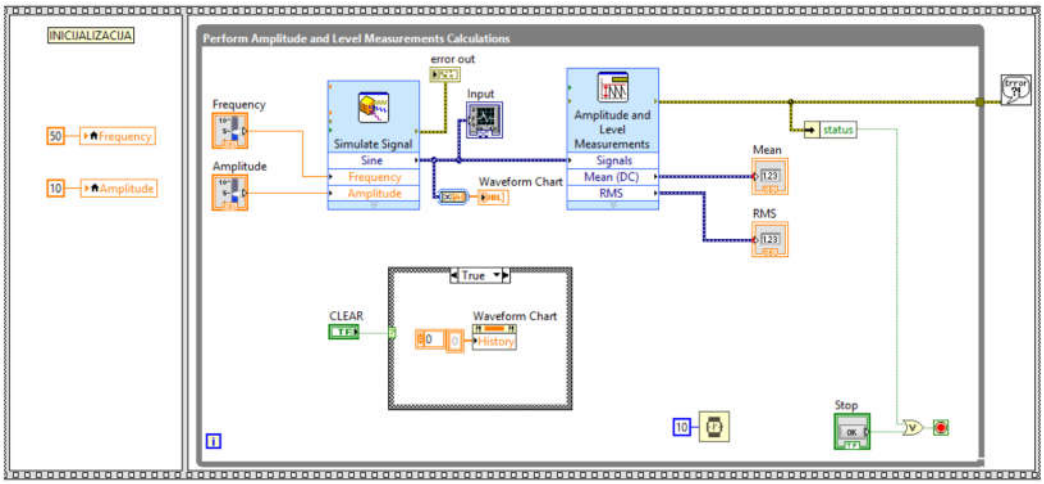


Sl. 3.5.1. Kreiranje „*error*“ klastera



Sl. 3.5.2. „*Error*“ klaster, izgled na *Front Panel*-u

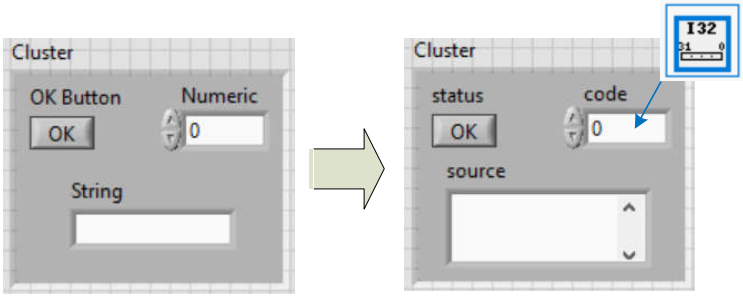
4. Klasteri se mogu „raspakovati“ pomoću funkcije ***Functions»Programming» Cluster, Class & Variant»Unbundle by Name.*** Funkcija *Unbundle by Name* se može „razvući“ tako da ima onoliko izlaza koliko klaster ima tipova podataka u sebi.
5. Modifikovati program u skladu sa Sl. 3.5.3. tako da *status* promenljiva *error* klastera bude preko ILI kola dovedena na uslovni terminal *While* petlje.
6. **Sačuvati** izmene.



Sl. 3.5.3. „Raspakivanje“ *error* klastera i gašenje programa kada *status* greške postane TRUE

Zadatak 3.5.2.
Kreirati *error* klaster kontrolu i *error* klaster indikator. Omogućiti da se samo vrednost *status* promenljive *error* klaster indikatora menja pomoću logičke kontrole sa *Front Panel*-a, a da se *code* i *source* promenljive *error* klaster kontrole „preslikavaju“ u *code* i *source* promenljive *error* klaster indikatora.

Uputstvo:
1. Otvoriti nov **.vi** program.
2. Iz palete ***Controls»Modern»Array, Matrix & Cluster*** selektovati kontrolu *Cluster* i postaviti je na *Front Panel*. Potom u *Cluster* kontrolu na *Front Panel*-u postaviti sledeće tri kontrole PREMA DATOM REDOSLEDU: 1) *String Control* (***Controls»Modern»String & Path***), 2) *Numeric Control* (***Controls»Modern» Numeric***) i 3) *OK Button* (***Controls»Modern»Boolean***). Za *Numeric Control* podesiti da labela bude *code*, a da tip podataka bude *integer* 32-bita (desni klik miša na kontrolu, opcija *Representation, I32*) jer se tako standardno definiše u *error* klasteru. Takođe, podesiti da labele za *String Control*-u i *OK Button* budu *source* i *status*, respektivno, Sl. 3.5.4.

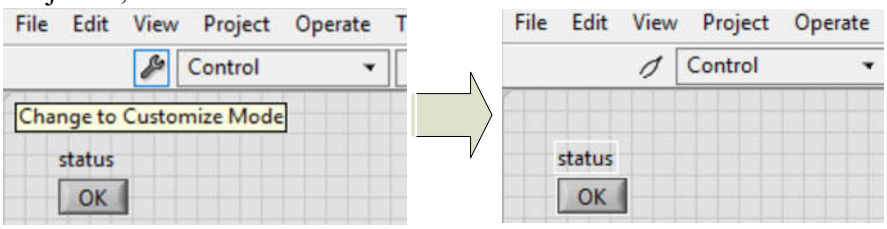


Sl. 3.5.4. Manuelno napravljen *error* klaster sastavljen iz tri promenljive različitog tipa

NAPOMENA: Vertikalni *scrollbar* kontrole *string* je onemogućen (*disabled*) sve dok se *string* kontrola ne proširi bar na dva reda. Kada se *string* kontrola proširi po vertikali tako da zahvata bar dva reda, onda postaje omogućena opcija *Vertical Scrollbar* (desni klik miša, *Visible Items*).

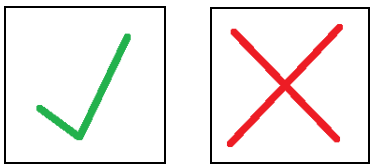
NAPOMENA: *Status* promenljiva u *error* klasteru trenutno nema standardni izgled (,). Promena izgleda kontrole/indikatora se može uraditi tako što se na kontrolu/indikator klikne desnim mišem i izabere opcija *Advanced/Customize*.

3. Kliknuti desnim mišem na *OK Button error* klastera i izabrati opciju *Advanced/Customize*. Otvoriće se novi prozor u kome treba izabrati môd za editovanje , Sl. 3.5.5.



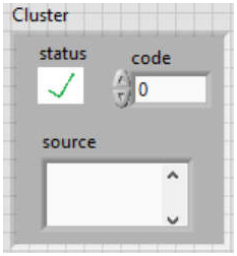
Sl. 3.5.5. Selekcija môda za editovanje kontrole/indikatora

4. U programu za kreiranje slika (npr. *Paint*-u) napraviti dve slike istih dimenzija kao na Sl.3.5.6.



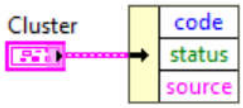
Sl. 3.5.6. Slike za TRUE i FALSE stanja *status* promenljive *error* klastera

5. Kliknuti desnim mišem na kontrolu *status* u prozoru *Customize mode*. Izabrati opciju *Picture Item* i selektovati najpre izgled kontrole za TRUE stanje . Kliknuti desnim mišem na kontrolu i izabrati opciju *Import from file...* Selektovati fajl sa novom slikom za TRUE stanje . Sličan postupak za pomenu izgleda u ponoviti za FALSE stanje. S obzirom na to da standardna *status* promenljiva u *error* klasteru nema natpis OK na samom dugmetu, izbrisati natpis OK korišćenjem *Labeling Tool*-a .
6. Sačuvati editovanu kontrolu izborom opcije ***File»Save*** kao *status_kontrola.ctl* fajl**.** Zatvoriti prozor za editovanje kontrola, a za pitanje „*Replace the original control „status“ with „status_kontrola.ctl?*“ kliknuti na opciju *Yes*. Novi izgled *error* klastera je prikazan na Sl. 3.5.7.



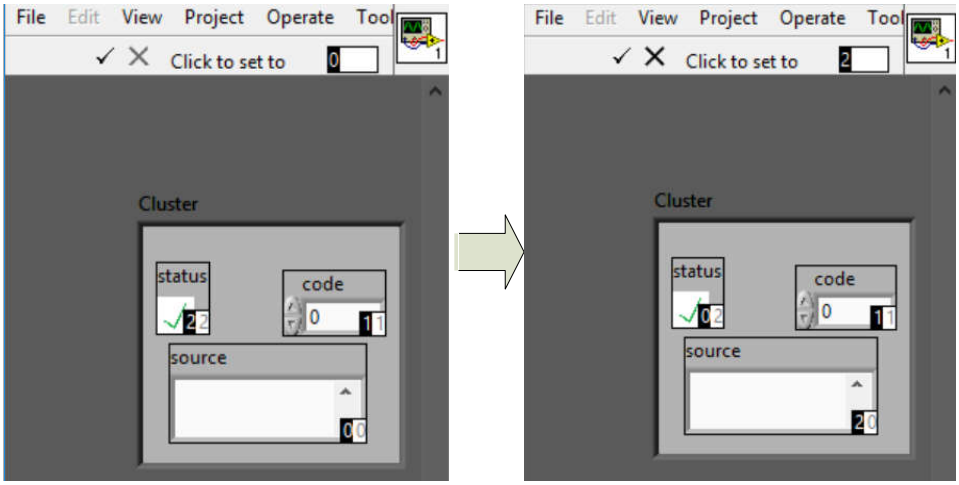
Sl. 3.5.7. Izgled editovanog *error* klastera

7. U *Block Diagram*-u izabrati funkciju ***Functions»Programming» Cluster, Class & Variant»Unbundle by Name*** i povezati je na *error* klaster. Proširiti funkciju *Unbundle by Name* tako da ima tri izlaza (pomoću *Position/Size/Select* alatke ), Sl. 3.5.8. Uočiti da je redosled promenljivih nestandardni, tj. da je prva promenljiva *code*, druga *status*, treća *source* (standardni redosled je: *status, code, source*). Ovakav redosled je nastao zbog redosleda postavljanja kontrola u klaster u tački 2.



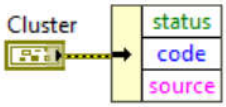
Sl. 3.5.8. „Raspakivanje“ editovanog error klastera – nestandardni redosled promenljivih u njemu

8. Promeniti redosled promenljivih u standardni. Najpre izabrati *Position/Size/Select* alatka  , potom kliknuti desnim mišem na *error* klaster kontrolu i izabrati opciju *Reorder controls in cluster*. Otvoriće se prozor za editovanje redosleda u kome se redosled promenljivih u klasteru zadaje redosledom kliktanja mišem po kontrolama, Sl. 3.5.9. Kliknuti sledećim redom na: 1) *status* kontrolu, 2) *code* kontrolu i 3) *source* kontrolu. Prihvatiti novi redosled promenljivih klikom na opciju *Confirm*  u *Status Toolbar*-u.



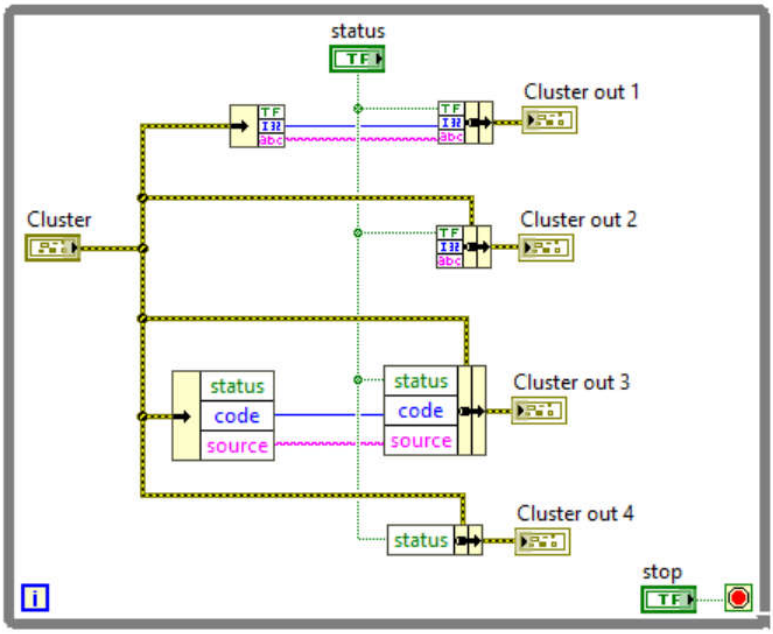
Sl. 3.5.9. Promena redosleda promenljivih u *error* klasteru

9. Izbrisati *Unbundle by name* funkciju iz tačke 7, potom uneti novu funkciju *Unbundle by name* i povezati je sa *error* klasterom koji sada ima novi redosled promenljivih. Uočiti da je sada redosled promenljivih na izlazu *Unbundle by name* funkcije u skladu sa novim redosledom promenljivih u klasteru, Sl. 3.5.10.

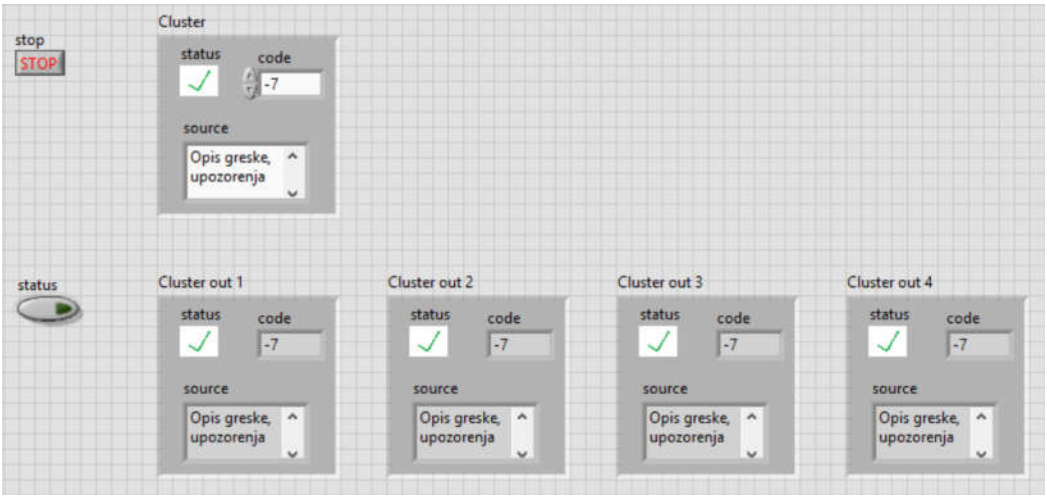


Sl. 3.5.10.

10. Iskopirati *error* klaster kontrolu (selektovati je, pritisnuti CTRL+C na tastaturi, a potom CTRL+V). Iskopiranu kontrolu „pretvoriti“ u *error* klaster indikator (selektovati kontrolu, desni klik miša, opcija *Change to Indicator*).
11. Kreirati *Block Diagram* kao na Sl. 3.5.11. i *Front Panel* kao na Sl. 3.5.12. koristeći *error* klaster kontrolu, 4 *error* klaster indikatora, *Unbundle* funkciju, *Unbundle by name* funkciju i logičke kontrole *status* i *stop*.



Sl. 3.5.11.



Sl. 3.5.12.

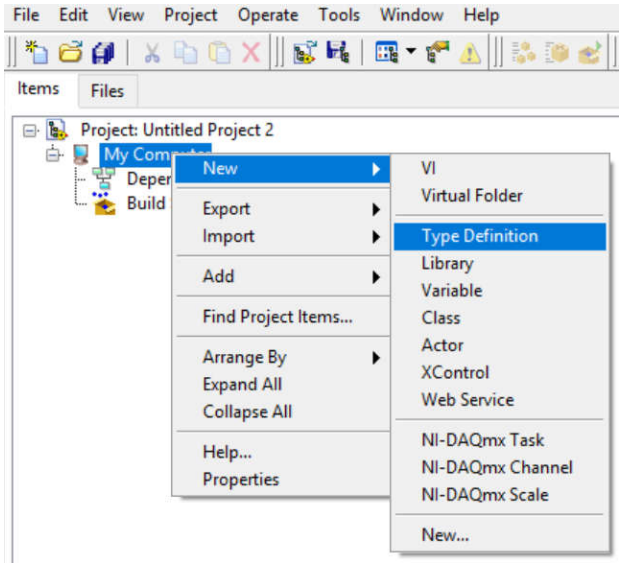
12. Pokrenuti program i uočiti kako se promenom stanja logičke kontrole *status* 

menja stanje promenljive *status* u *error* klaster indikatorima. Uočiti i da se vrednosti promenljivih *code* i *source* iz *error* klaster kontrole preslikavaju u odgovarajuće vrednosti *error* klaster indikatora.
13. Zaustaviti program i sačuvati ga kao *03_03_Error_klaster.vi*.

3.6 Striktno definisanje kontrola/indikatora

Ukoliko u programskom kôdu na više mesta postoje kontrole/indikatori koji treba da budu identičnih osobina, preporuka je da se najpre napravi njihov „templejt“, a da se onda na sva mesta instance tog templejta postavljaju. „Templejt“ može biti manje striktan (*type definition*), tj. može da omogući da samo tip podataka bude zajednički svima, a može biti i u potpunosti struktan (*strict type definition*), tj. može da zahteva da svim instancama kontrola/indikatora izgled bude identičan (da osim istog tipa podataka imaju i istu boju, veličinu i stil).

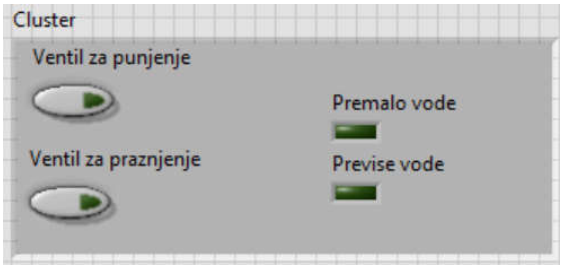
1. Otvoriti novu projektnu datoteku, ***File»Create Project»Blank Project***.
2. Pozicionirati miša iznad opcije *My Computer*, uraditi desni klik miša i odabrati opciju ***New»Type definition***. Otvoriće se prozor za definisanje kontrole/indikatora.
3. Postaviti u novootvorenom prozoru *Enum* kontrolu. Dizajnirati je po želji (boja, veličina, stila) i obavezno definisati i nekoliko vrednosti u padajućoj listi *Enum* kontrole (desni klik miša, *Edit Items*). Na *Status Toolbar*-u padajućem meniju izabrati „*type definition*“.
4. Sačuvati kontrolu kao *Enum.ctl* fajl.
5. Otvoriti novi .vi u okviru tekućeg projekta. Na njegov *Front Panel* „odvući“ *Enum.ctl* fajl iz stabla projekta. Uočiti da Enum kontrola koja je nastala na ovakav način više nema opciju *Edit Items*, ali da se veličina, boja i stil kontrole mogu menjati.
6. Otvoriti ponovo *Enum.ctl* fajl iz stabla projekta. Na *Status Toolbar*-u padajućem meniju izabrati ovaj put „*strict type definition*“ i sačuvati Enum.ctl fajl.
7. *Enum* kontrola koja je izvučena na novom .vi i dalje nema *Edit Items* opciju, ali su joj takođe zaključane i opcije za promenu veličine, boje i stila.



Sl. 3.6.1.

3.7 Zadaci za samostalni rad

Zadatak 3.7.1.
Napraviti program koji omogućava kontrolu nivoa vode u rezervoaru zapremine 1000 litara. Rezervoar poseduje ventil za punjenje i ventil za pražnjenje koji se mogu kontrolisati sa korisničkog interfejsa. Kada je otvoren ventil za punjenje, u rezervoar ulazi litar tečnosti na svakih 10 ms (pri svakoj iteraciji while petlje (koja traje 10 ms) povećati vrednost u rezervoaru za 1). Kada je otvoren ventil za pražnjenje, na svakih 10 ms iz rezervoara izlazi 2 litra tečnosti (pri svakoj iteraciji while petlje smanjiti vrednost u rezervoaru za 2). Ukoliko nivo vode pređe 990 litara, uključiti lampicu koja označava opasno visok nivo vode i ugasiti ventil za punjenje. Ukoliko nivo vode padne ispod 10 litara, uključiti lampicu koja označava opasno nizak nivo vode i ugasiti ventil za pražnjenje. Kontrole ventila i indikatore lampica grupisati zajedno u klaster kako bi se obezbedilo olakšano uključivanje još rezervoara u budućnosti. Pri pokretanju programa, nivo vode je 100 litara i ventili su zatvoreni. Pri zaustavljanju programa obezbediti da su oba ventila zatvorena.
Napomena: koristiti klaster definisan kao na Sl. 3.7.1.



Sl. 3.7.1.

Zadatak 3.7.2.
Napraviti program za igru na sreću. U ovoj igri na sreću povlačenjem ručice na korisničkom interfejsu na dole u članove tročlanog niza brojeva počinju da se upisuju slučajne celobrojne vrednosti od 0 do 100. Obezbediti da se vrednosti menjaju na 2 ms. Vraćanjem ručice u početni položaj se upisivanje vrednosti prekida i trenutno zatečeni brojevi se sabiraju. Ukoliko je rezultat sabiranja veći ili jednak 150, igrač dobija poen. U suprotnom igrač gubi dva poena. Igra se završava kada igrač odluči da unovči svoje poene ili kada izgubi sve poene. Početni broj poena pri pokretanju igre je 5.
Dodatni zahtevi:

- Obavestiti igrača da je izgubio sve poene pre isključenja programa.
- Ukoliko igrač pritisne dugme “Unovči”, dati mu mogućnost da se predomisli.