INSTRUMENTATIE VIRTUALA

CURS 5

Clusteri

Structuri de programare in LabVIEW

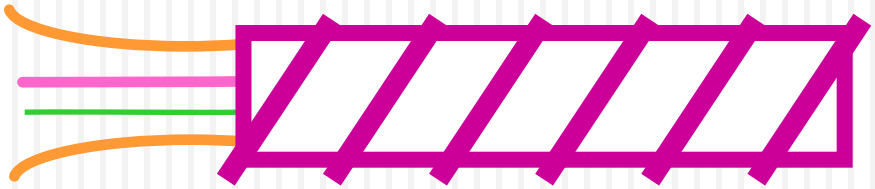
Obiective

- Utilizarea datelor de tip clusteri
- Utilizarea structurilor de programare decizionale

Clusteri (Clusters)

Clusteri

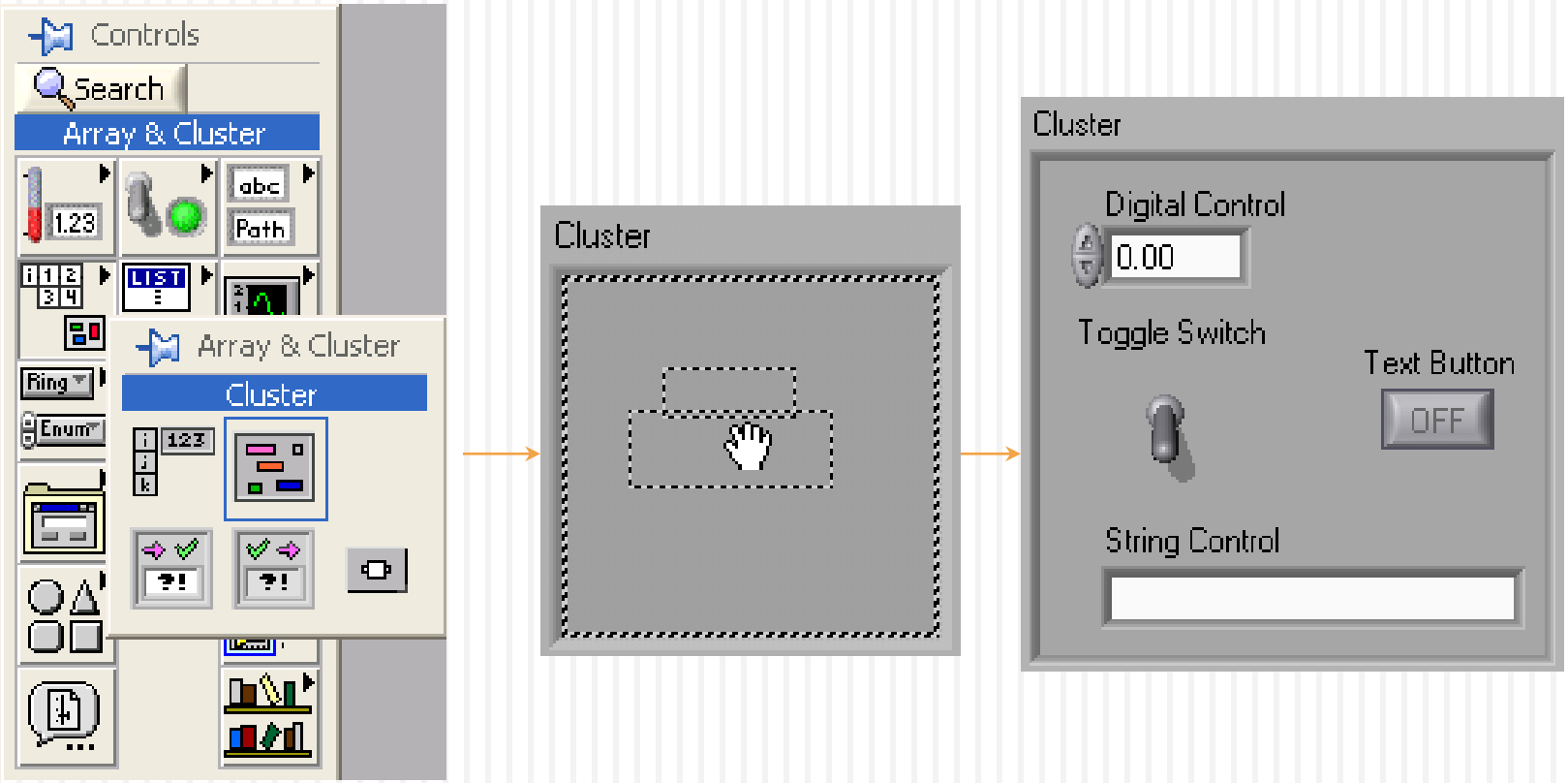
- **Clusterul** este o structura care aduna datele impreuna
- Datele pot fi **de tipuri diferite** !
- Este analog cu *struct* in C
- Elementele trebuie sa fie toate sau **controale** sau **indicatoare**
- **Clusterul** seamana cu un fascicul de fire dintr-un cablu !



Controale si indicatoare de Cluster

1. **Luati un Cluster din subpaleta Array & Cluster**

2. **Plasati in aceasta rama de Cluster obiectele dorite**



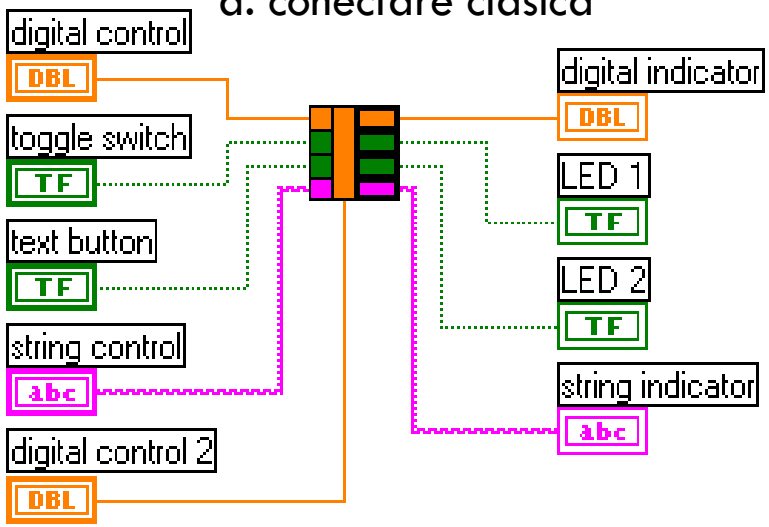
Folosirea Cluster-ului pentru a trimite date unui SubVIs

Printr-un Cluster se pot trimite simultan multe date unui fir (terminal) de intrare sau iesire intru-un Icon (SubVI)

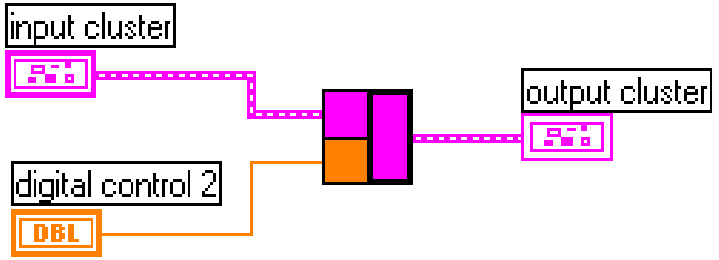
Se poate depasii astfel limitarea de 28 terminale la un conector !

Simplificam mult “cablarea”

a. conectare clasică



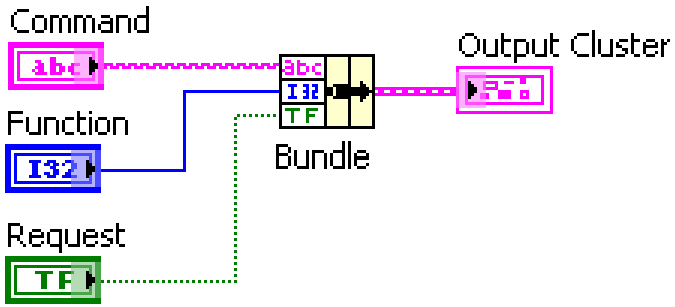
b. conectare cu clusteri



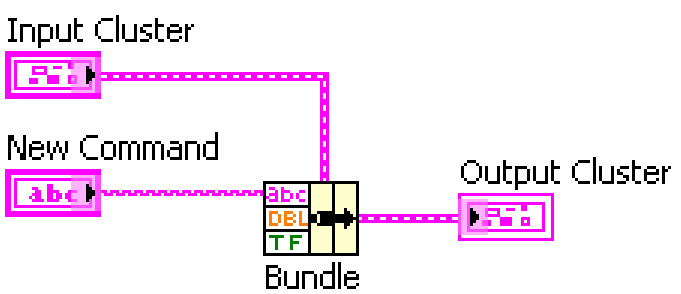
Functii de Cluster – functia Bundle

Bundle

Realizarea unui nou Cluster

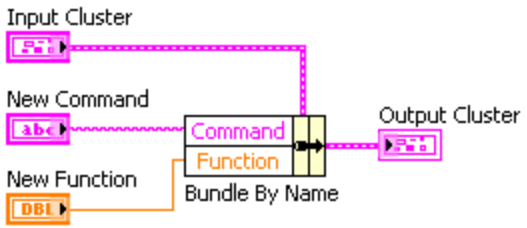


Modificarea unui cluster existent

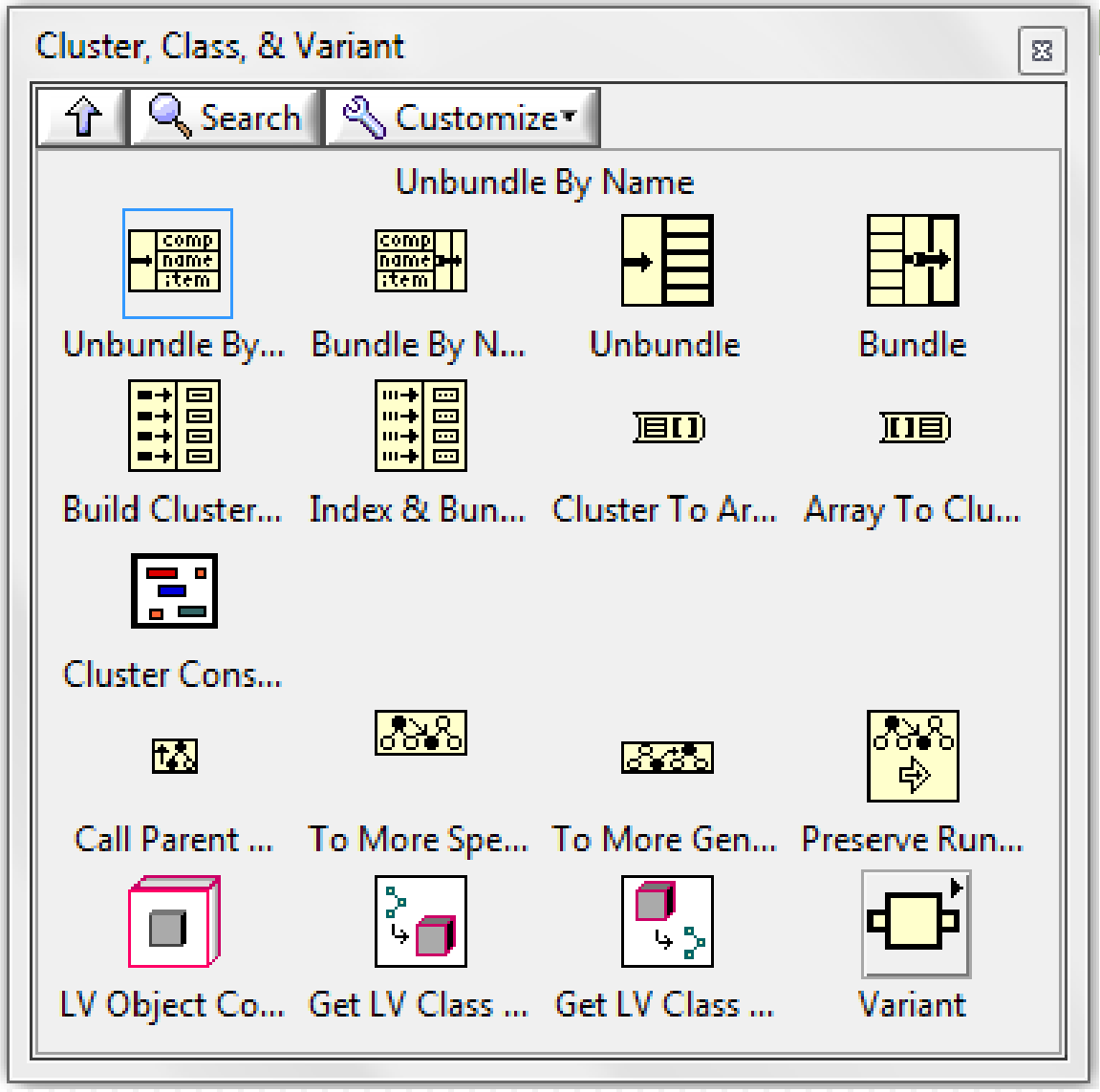


Bundle By Name

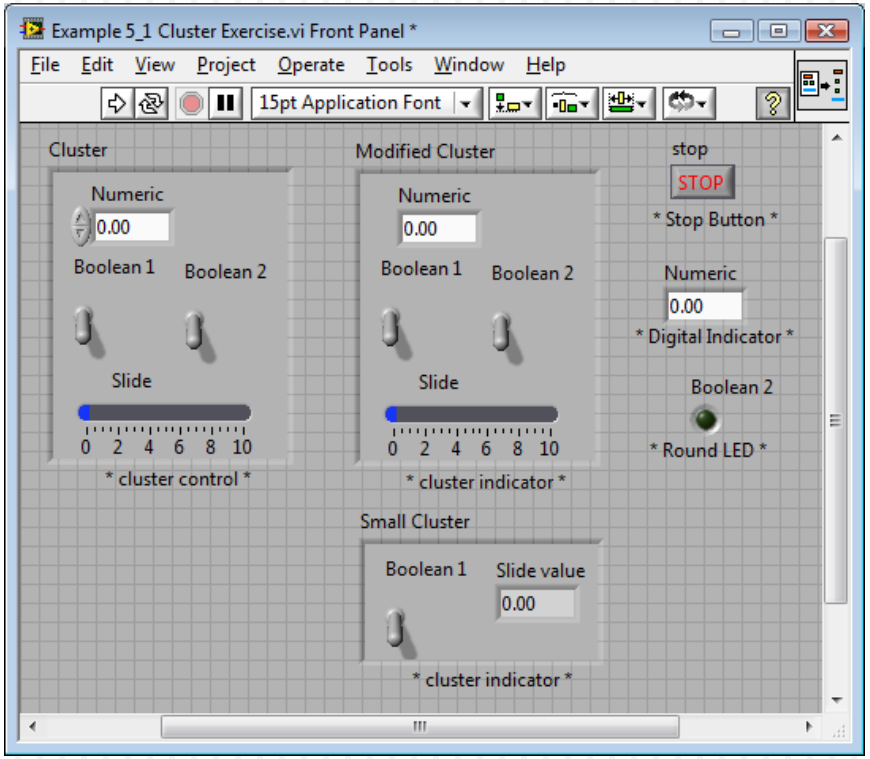
Trebuie sa existe un Cluster pentru a putea folosi aceasta functie.

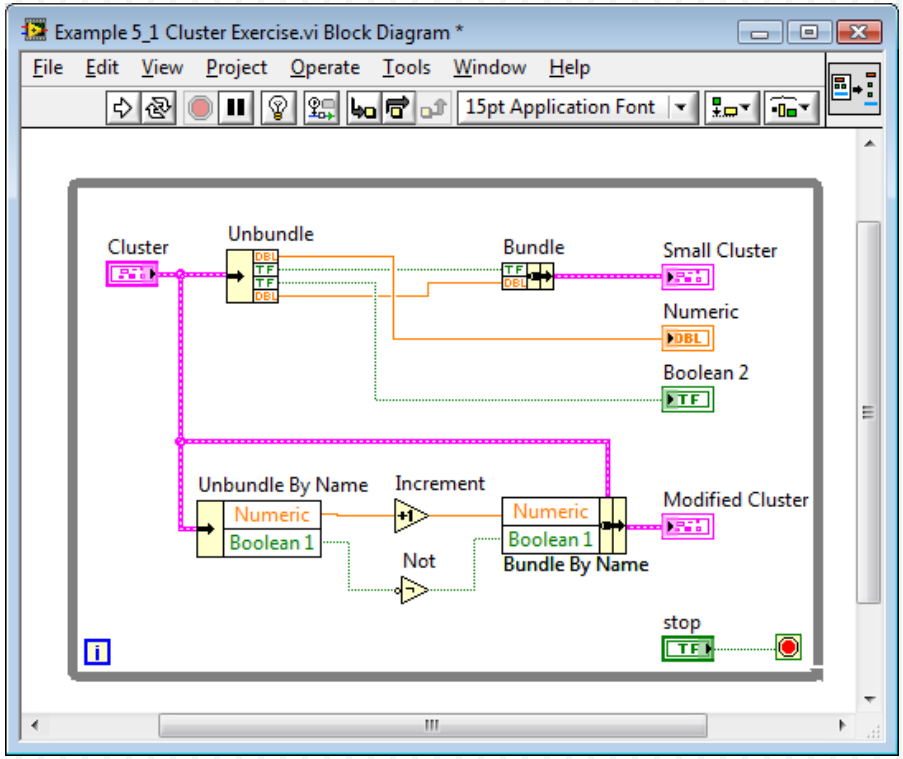


Functii de Cluster



Exemplu

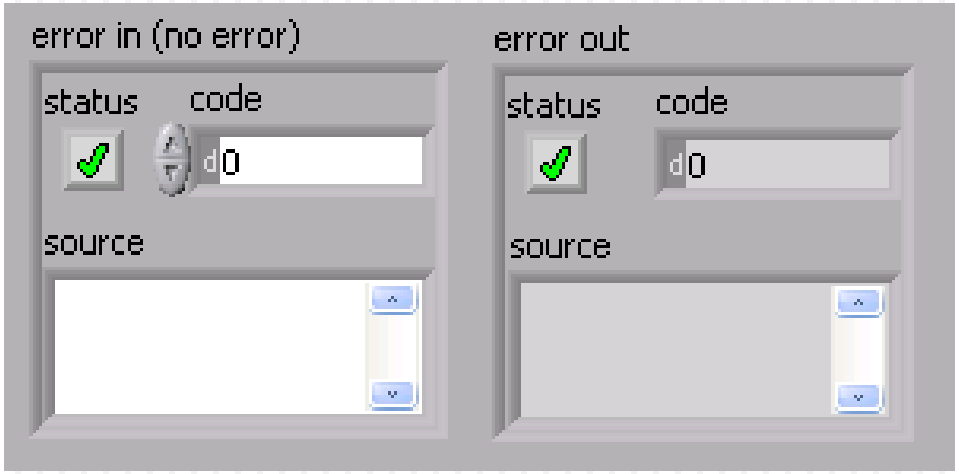

Example 5_1 Cluster Exercise.vi Front Panel *
File Edit View Project Operate Tools Window Help
15pt Application Font
Cluster
Numeric
0.00
Boolean 1
Boolean 2
Slide
0 2 4 6 8 10
* cluster control *
Modified Cluster
Numeric
0.00
Boolean 1
Boolean 2
Slide
0 2 4 6 8 10
* cluster indicator *
stop
STOP
* Stop Button *
Numeric
0.00
* Digital Indicator *
Boolean 2
* Round LED *
Small Cluster
Boolean 1
Slide value
0.00
* cluster indicator *

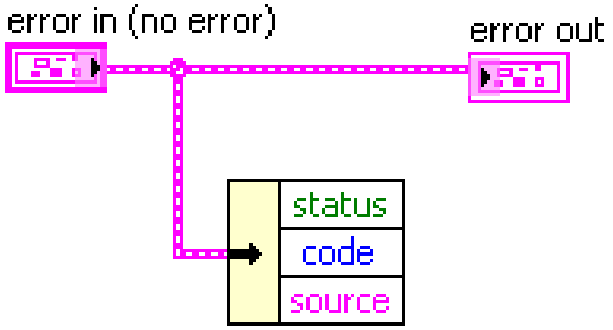

Example 5_1 Cluster Exercise.vi Block Diagram *
File Edit View Project Operate Tools Window Help
15pt Application Font
Cluster
Unbundle
Bundle
Small Cluster
Numeric
Boolean 2
Unbundle By Name
Numeric
Boolean 1
Increment
Not
Numeric
Boolean 1
Bundle By Name
Modified Cluster
stop

Cluster – pentru Erori

Prin cablarea clusterilor “error in” si “error out” in fiecare VI, realizati o gestiune a erorilor in noul VI

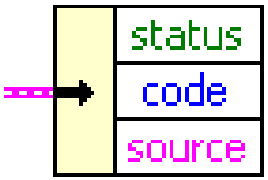
Clusterii de eroare localizati in paleta Controls»Array & Cluster include diverse componente de informare





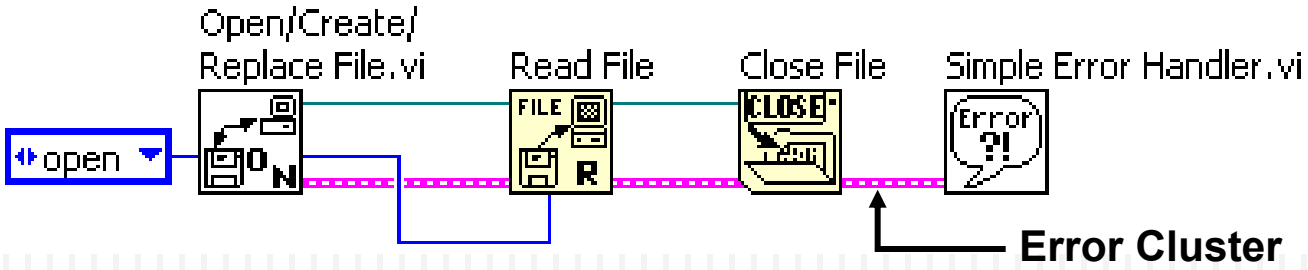
Detalii ale Cluster-ilor de Eroare

- **Status** este o valoare boleana care raporteaza TRUE daca a aparut o Eroare. Multe VI-uri, Functii si Structuri care accepta date boleene, recunosc, de asemenea acest parametru.
- **Code** este un “signed 32-bit integer” care da un numar de identificare al Erorii. Un cod de Eroare, diferit de zero, impreuna cu un semnal de FALSE A, este mai mult un semnal de alerta decit o Eroare.
- **Source** este un Sir (String) care identifica unde a aparut eroare.



Gestiunea Erorilor folosind Clusteri

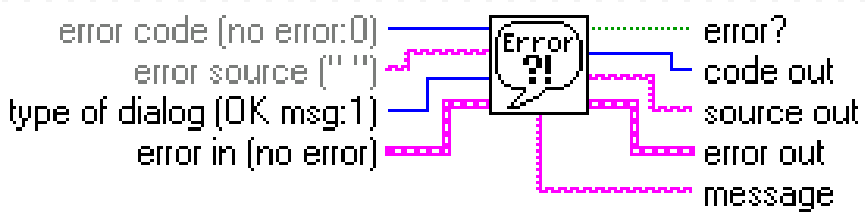
- Se poate monitoriza evolutia erorilor in “schema bloc” (Diagrama) programului realizat (VI-ului)
- Manevrarea erorilor in LabVIEW urmareste tot idea unui flux de date. Exact ca fluxul de date in VI-uri, avem si Fluxul de Erori legat de evolutia si transferul de erori de la un SubVI la altul
- Este recomandat sa cablati informatia de Eroare de la inceputul si pina la sfirsitul VI-ului



Simple Error Handler

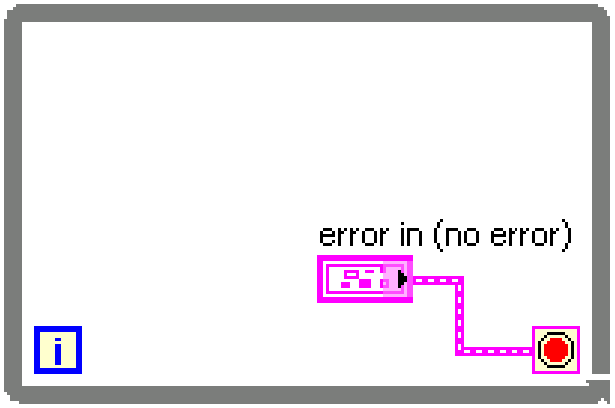
Folositi **Simple Error Handler** pentru a afisa eroarea la sfirsitul fluxului de executie (sfirsitul rularii VI-ului)

Iconul **Simple Error Handler** poate fi gasit in paleta **Functions»All Functions»Time and Dialog.** Acest icon se leaga cu terminalul **Error In (no error)** la Error out-ul ultimului SubVI



Folosirea buclei While Loops pentru Error Handling

- Se poate cabla clusterul de eroare la terminalul de conditionare al unei bucle While pentru a stopa iteratiile
- Numai starile TRUE sau FALSE ale parametrilor de eroare sunt transmisi terminalului de conditionare
- Cind apare o eroare bucla While este oprita



Structuri de programare (continuare)

Luarea unor decizii in VI

Subiecte

Realizarea de decizii cu functia Select

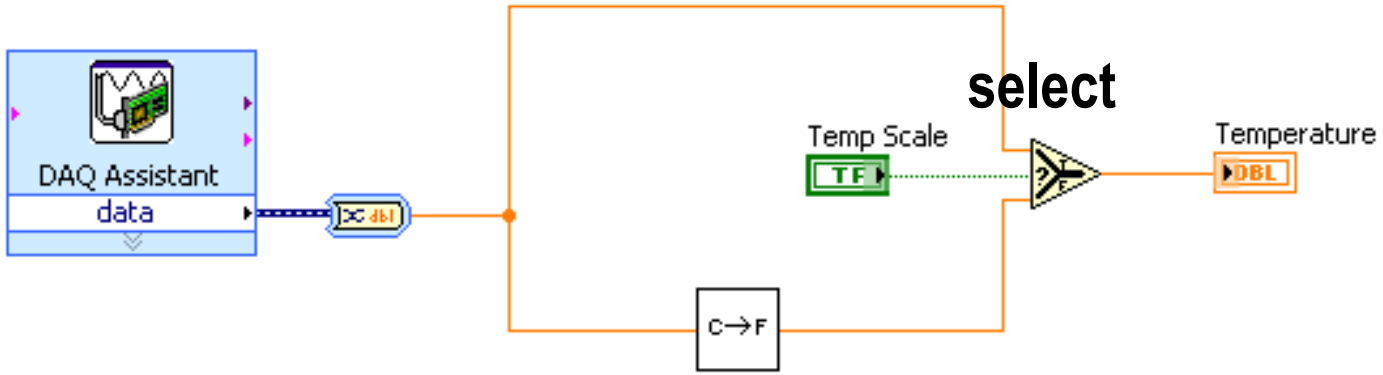
Structurile Case

Nodul de formule (Formula Nodes)

Sutructura SECVENTA

Decizie simpla: Functia Select

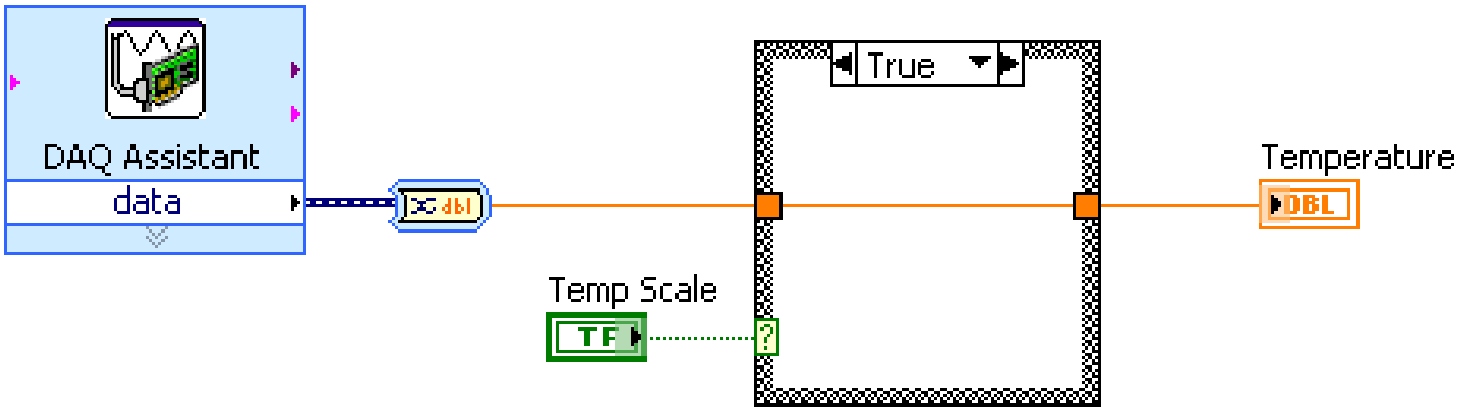
- Daca scala de temperatura este TRUE, treci pe sus;
  Daca scala de temperatura este FALSE, treci pe jos;



- Daca decizia ce trebuie luata este mult mai complexa ca functia Select trebuie sa luam o “Structura Case”

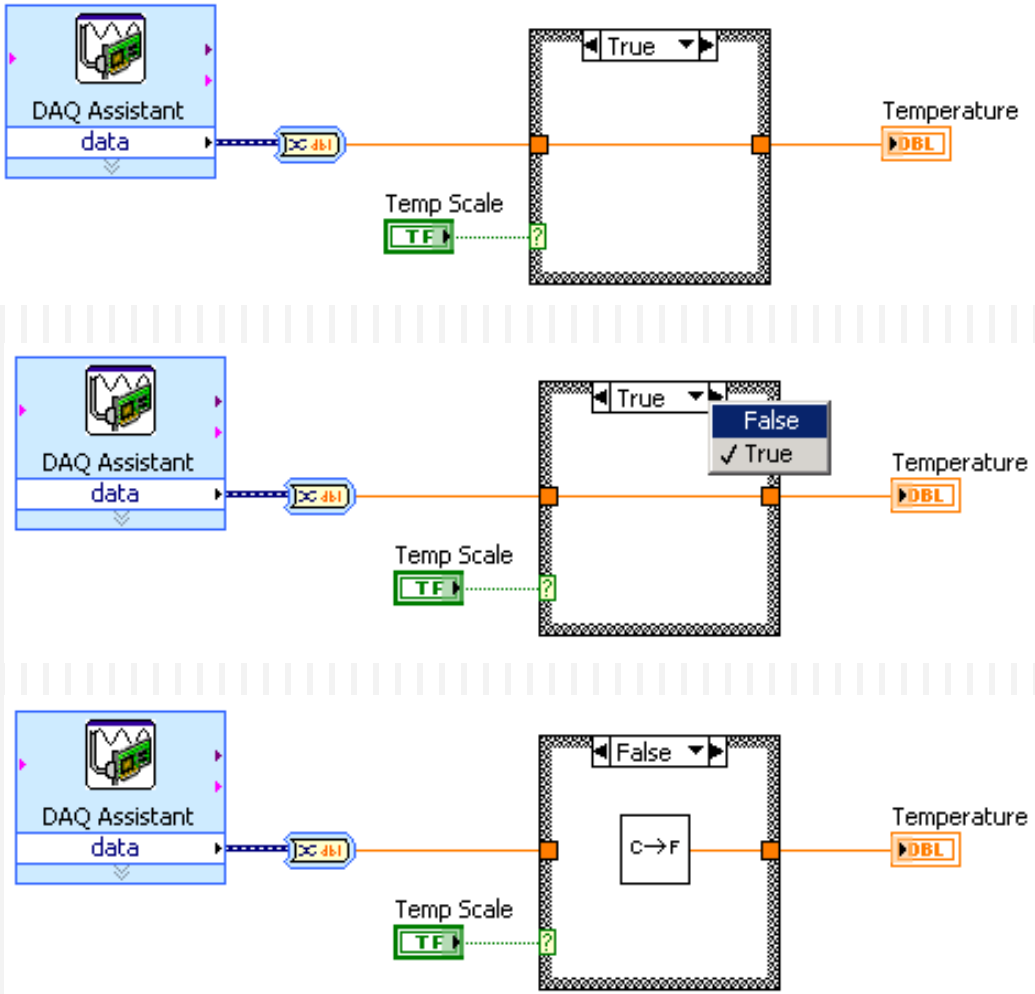
Structura Case

- Exemplu de structura Case boleana:
- Scala de temperatura TRUE, executa “True case”; scala de temperatura FALSE, executa “False case”.

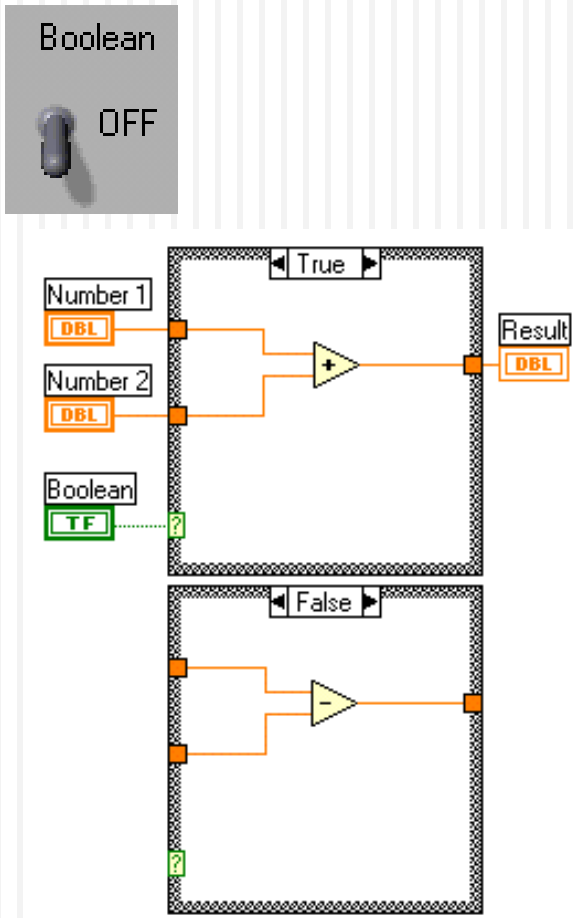


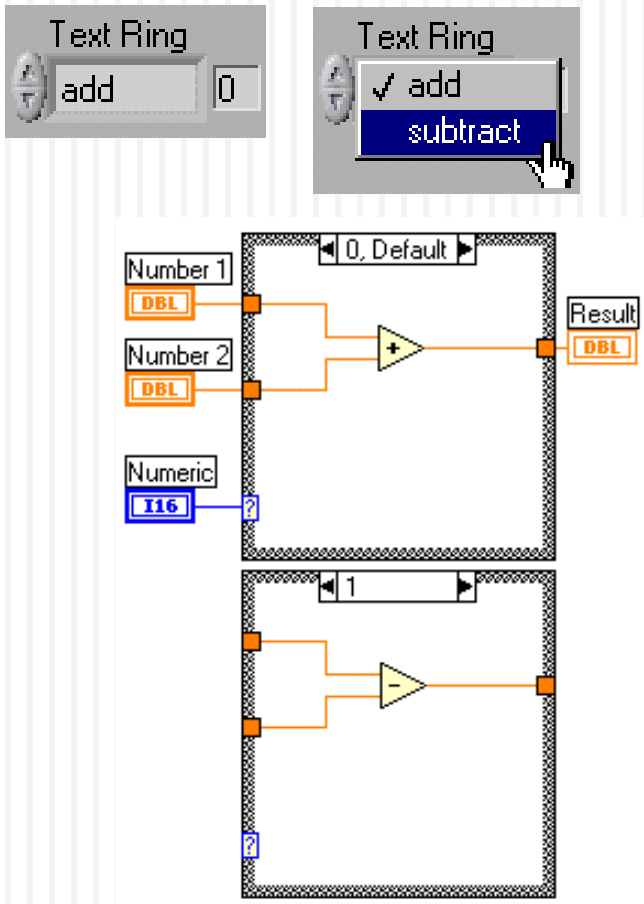
Structuri Case

- In subpaleta Strcuturi a paletei de Functii
- Introduceti iconuri (noduri) sau tragetile in structura
- Suprapuse ca un pachet de carti de joc, numai un Case vizibil la un moment dat



Structuri Case: Boolean si Numeric

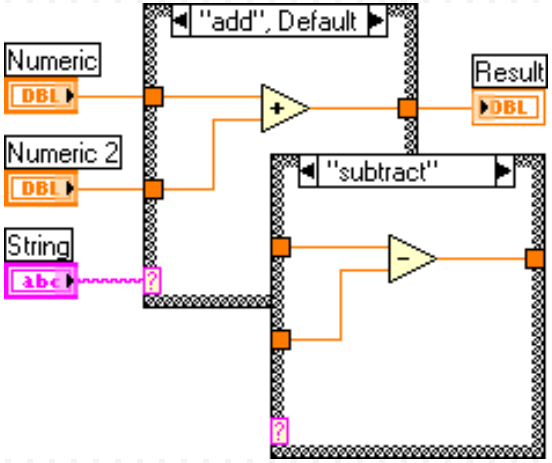




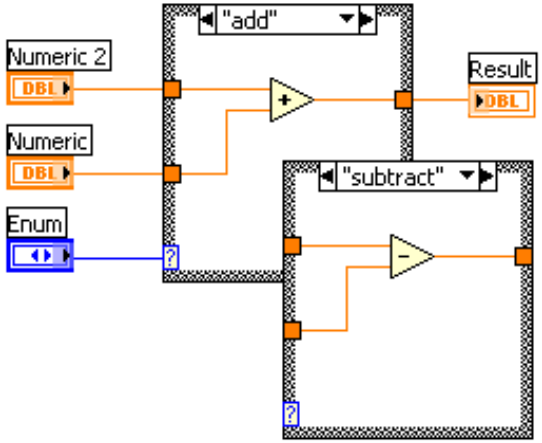
Cablati toate iesirile posibile in cazul Structurii Case

Structuri Case: String, Enum, Error

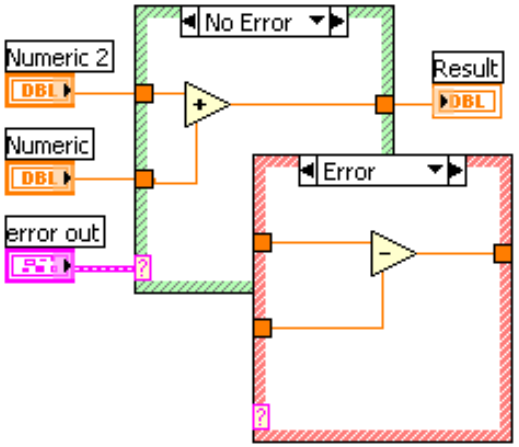
String Case



Enum Case



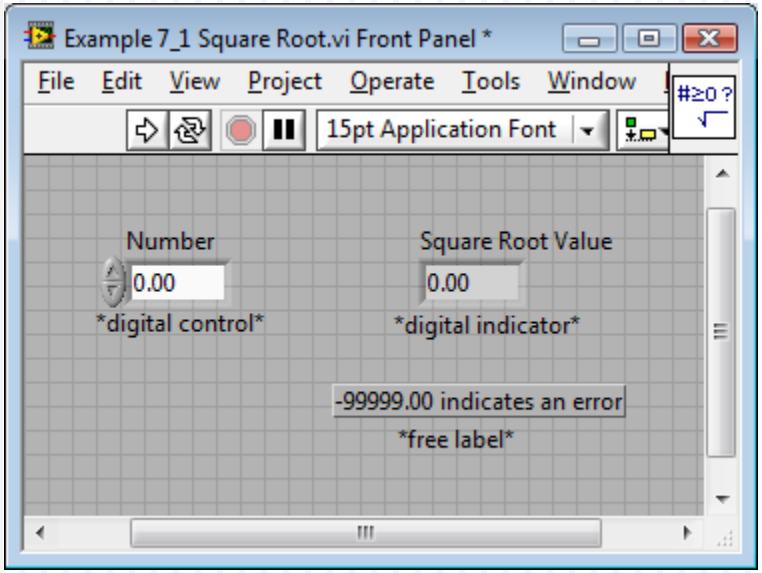
Error Case

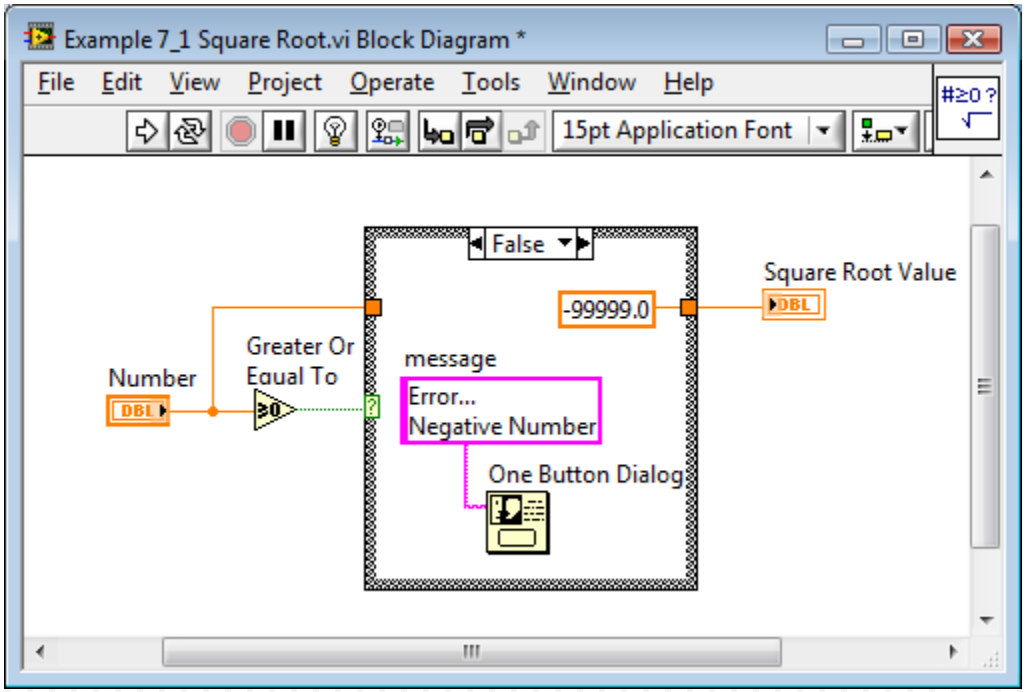


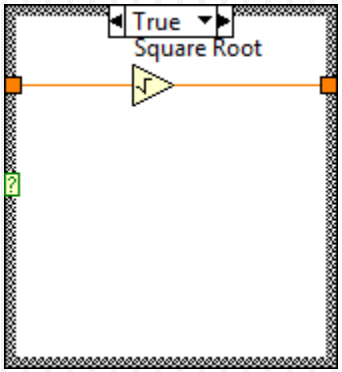
Exercitiul

Construiti un VI folosind Structura Case pentru a extrage un radical.

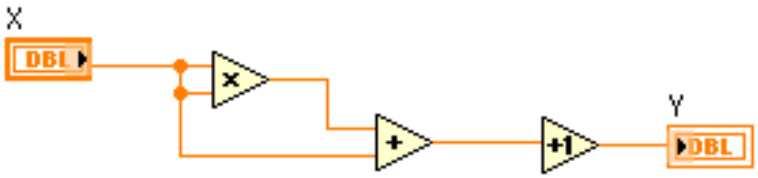
Square Root VI

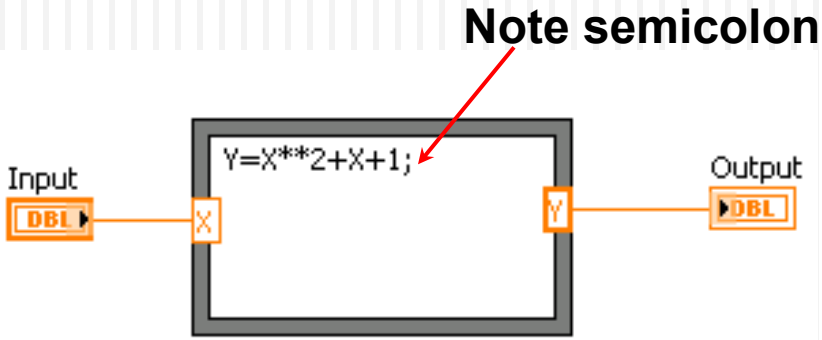






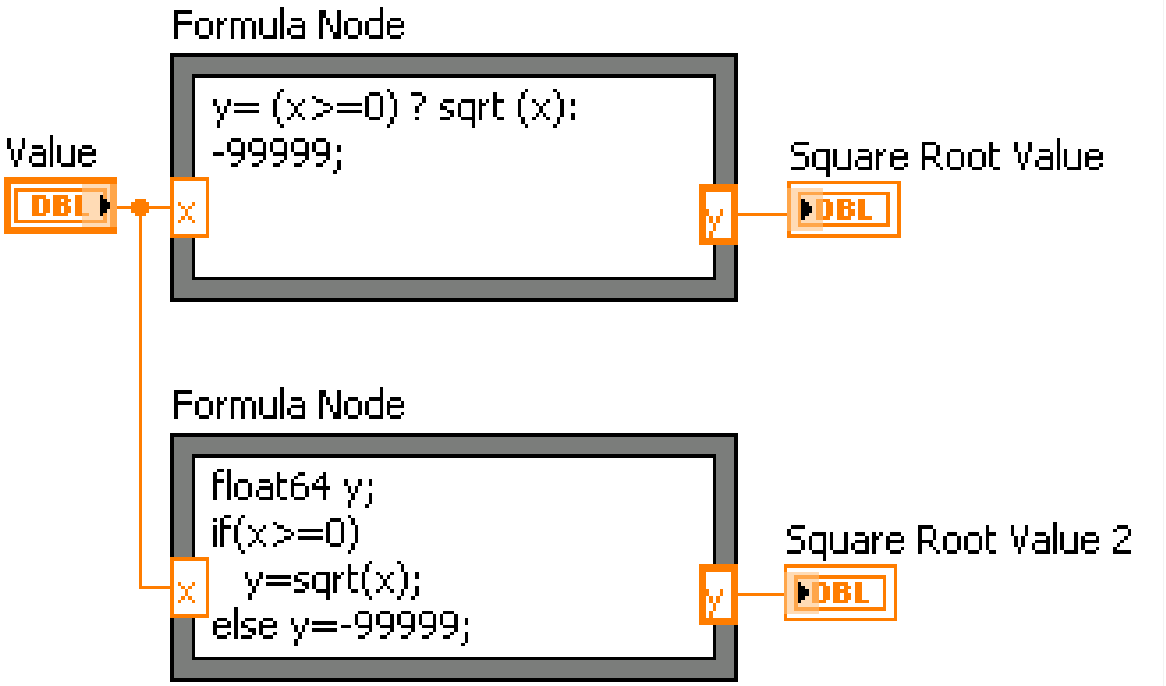
D. Nodul de formule: Formula Node





- In subpaleta Structuri a paletei de Functii
- Permite implementarea unor ecuatii complicate
- Variabile de intrare – iesire create pe periferie
- Numele variabilelor este “case sensitive"
- Fiecare ecuatie trebuie terminata cu (;)
- Helpul contextual ne arata functiile permise

Luarea deciziilor cu Nodul de Formule



- Doua cai diferite de implementare “if-then” in Formula Node
- Ambele structuri produc acelasi rezultat

Exercitiul

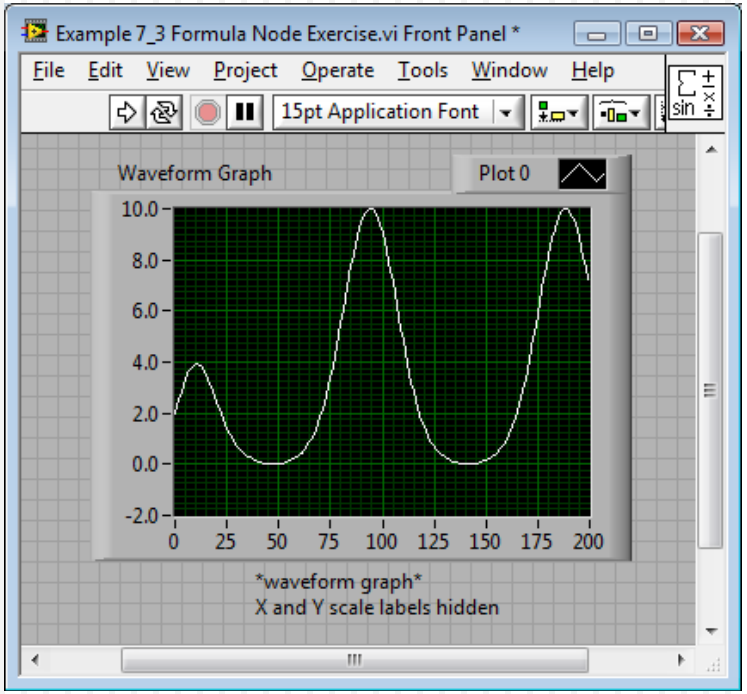
Formula Node Exercise VI

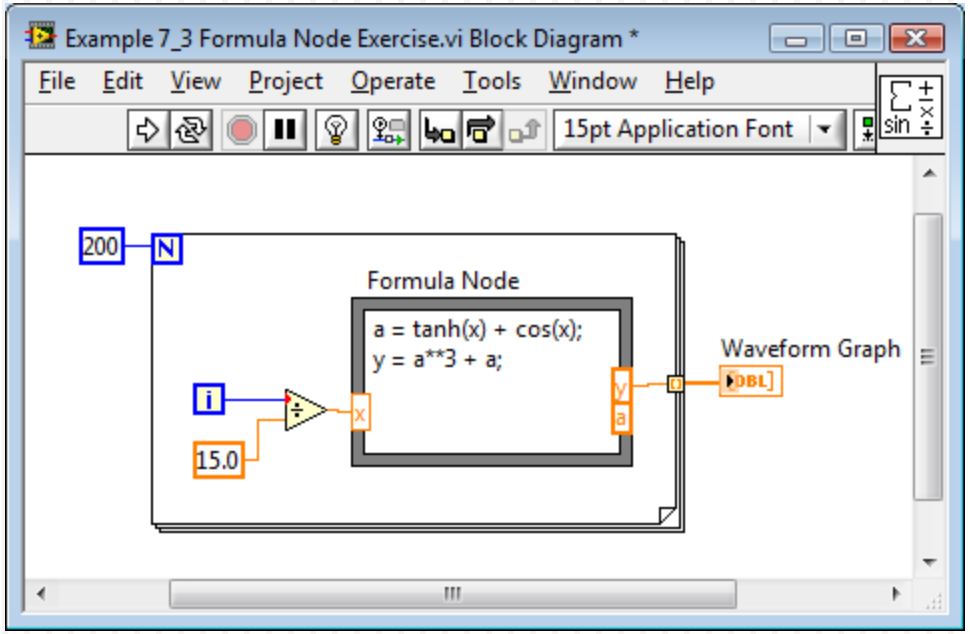
Folositi Nodul de Formule pentru a construi un VI

care produce operatii matematice complexe

si reprezinta grafic rezultatul.

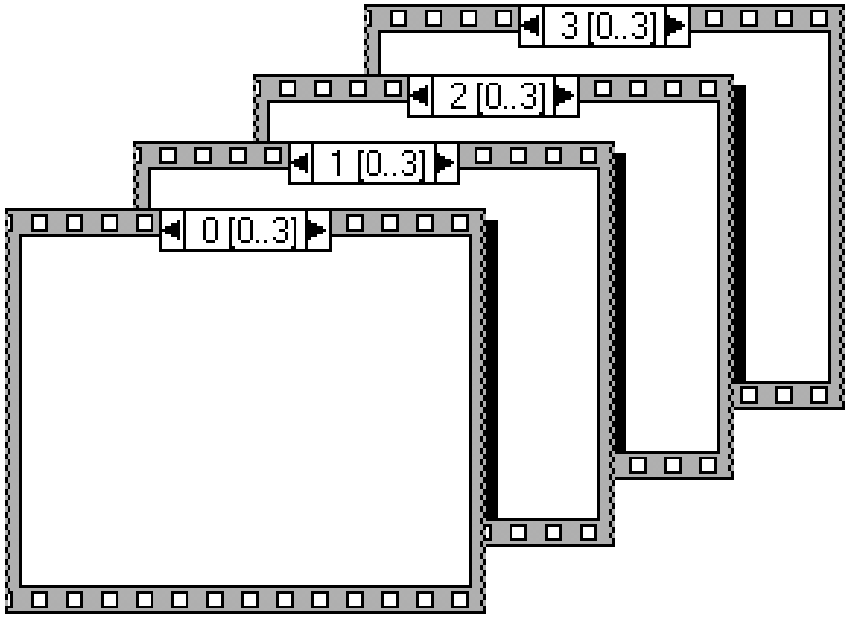
Exercitiul Formula Node Exercise VI



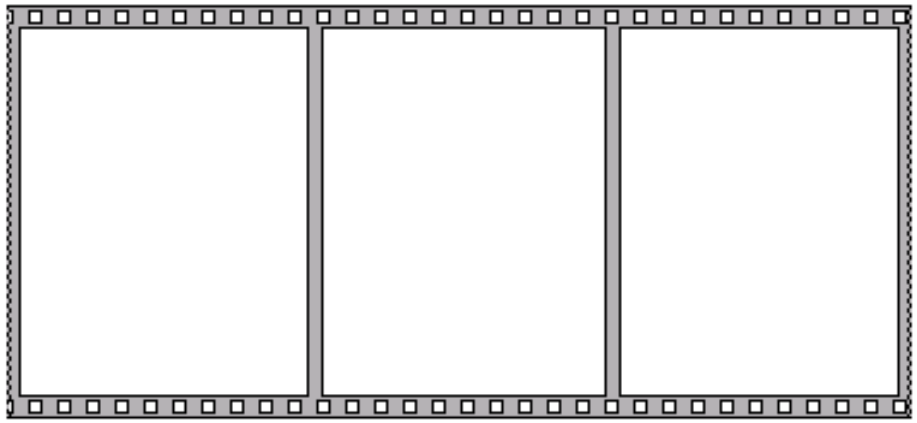


E. Sutructura SECVENTA

- Din subpaleta “Structures” a paletei de Functii
- Executa diagrama “secvential”, fereastra 0 (0..x), unde x este numarul total de ferestre



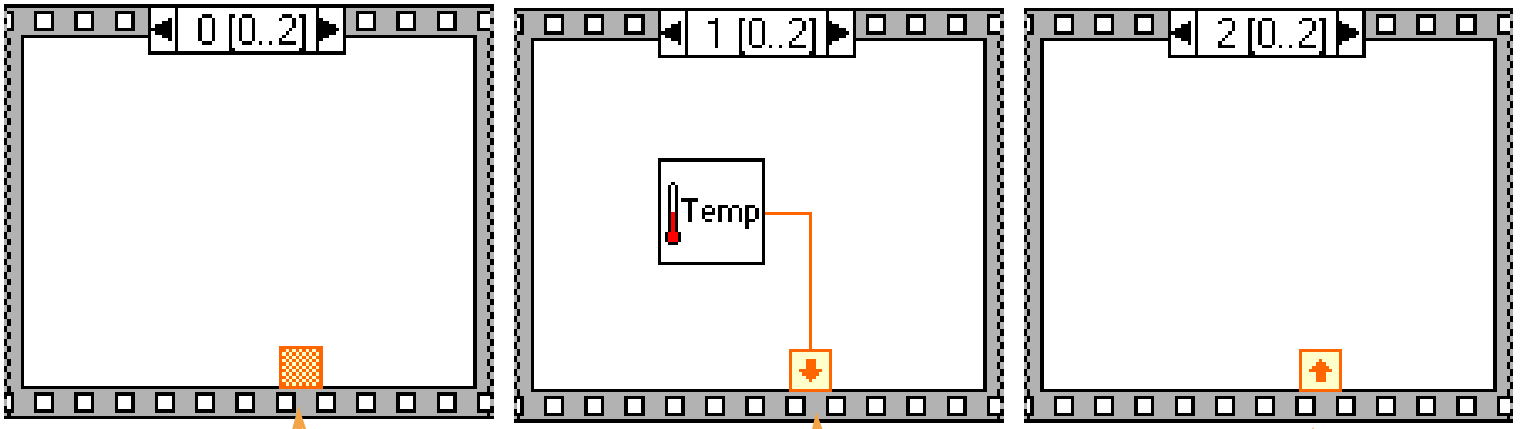
Stacked Sequence Structure



Flat Sequence Structure

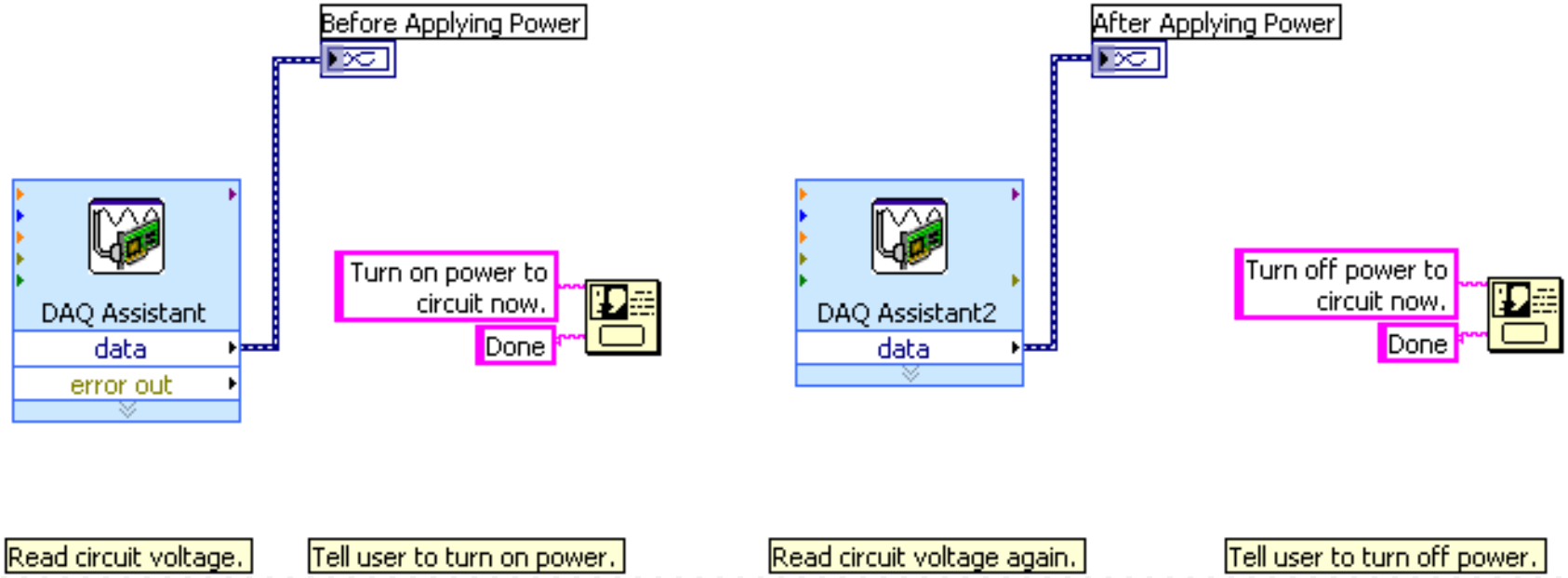
Cablare cu: Sequence Locals

- Trimite datele de la o fereastra la alta
- Realizata la periferia unei structuri de tip “Sequence”



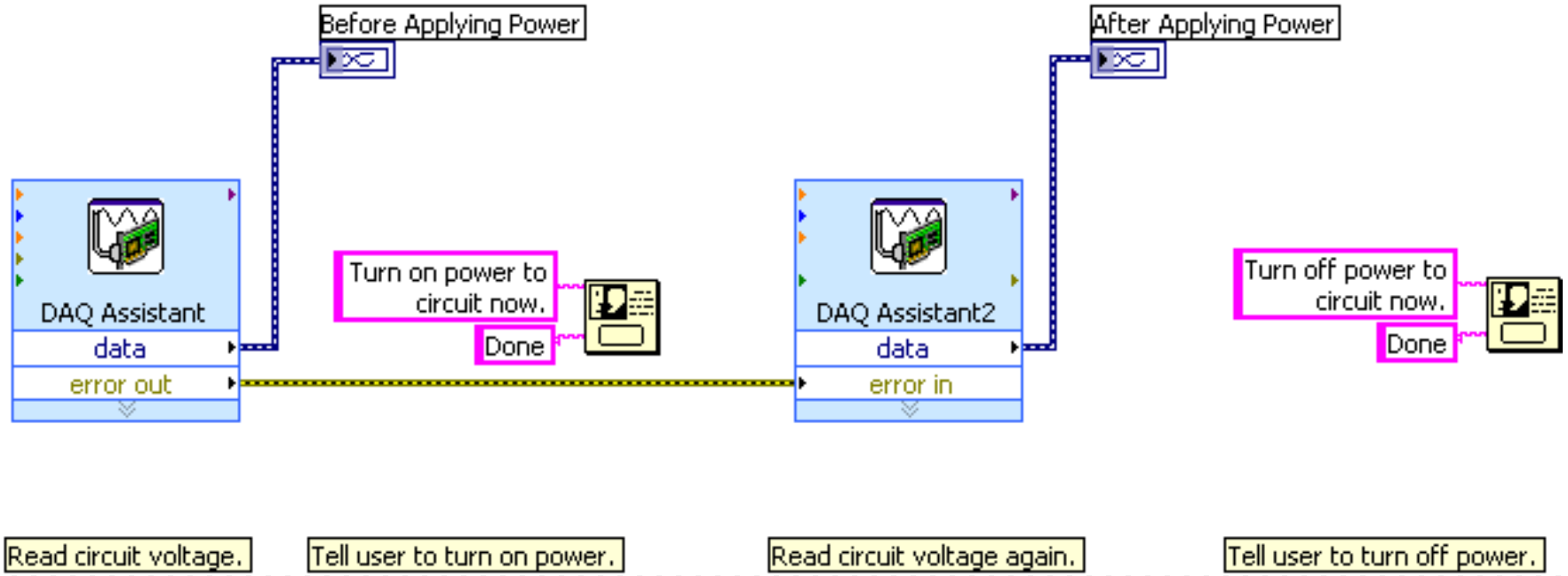
Programarea secventiala

- Multe VI-uri realizate necesita executarea secventiala a tascurilor
- In exemplul de mai jos nu exista nimic care sa forteze executia secventiala



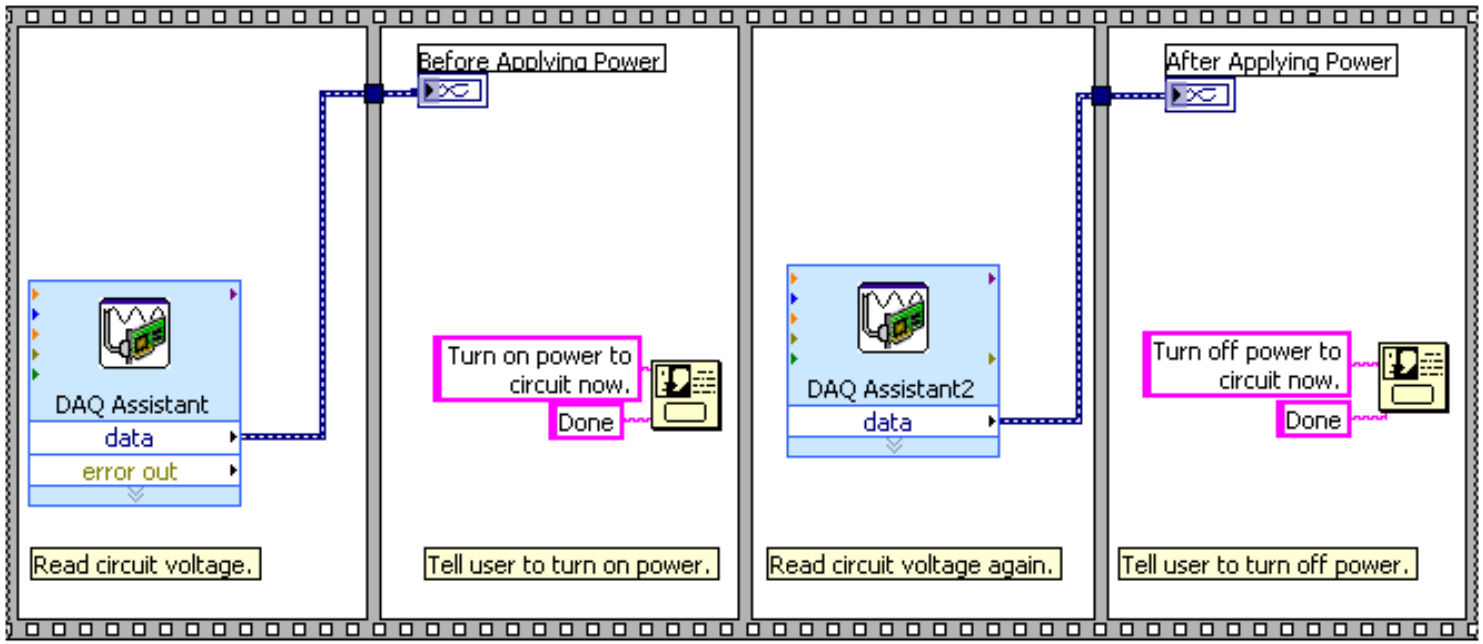
Programarea secventiala

Utilizarea error clusters pentru a forta ordinea de executie



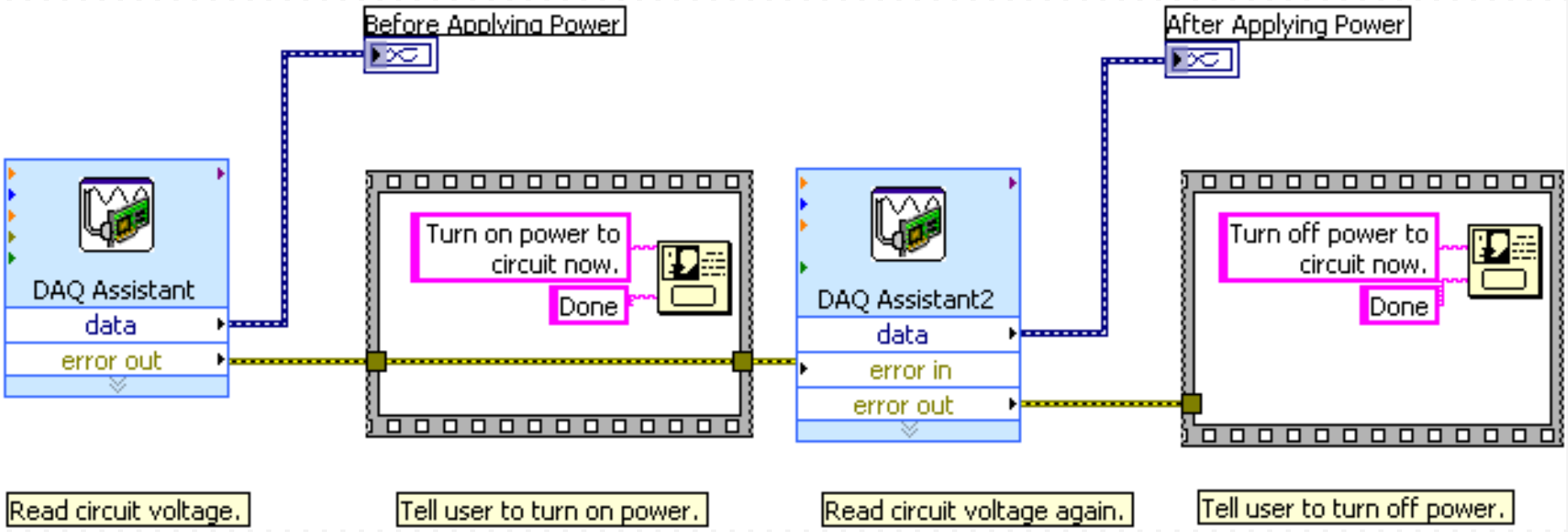
Programarea secventiala

Pentru a forta ordinea de executie, se utilizeaza structura Sequence



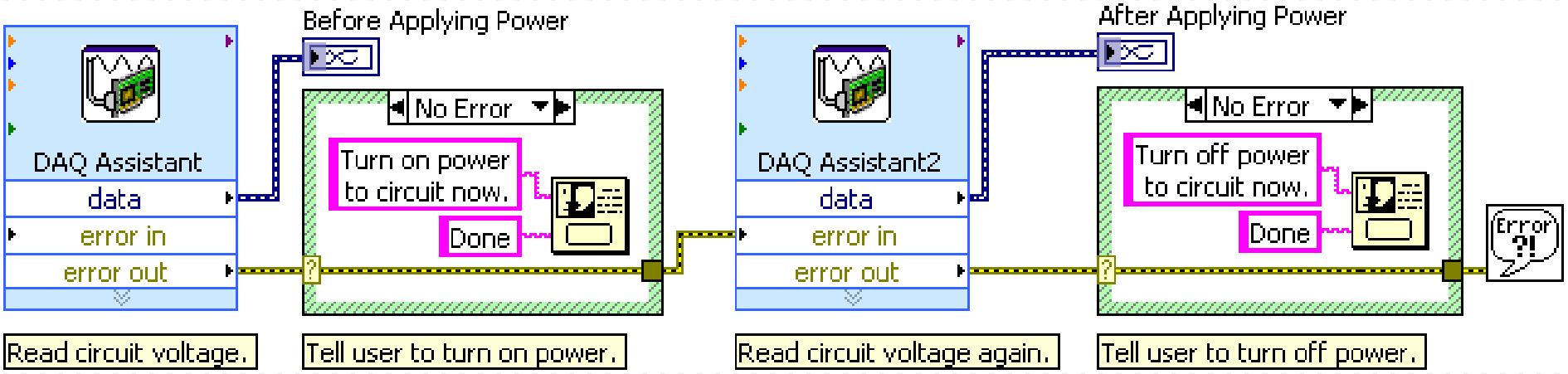
Programarea secventiala

- Trebuie evitata utilizarea exagerata a structurii Sequence
- Nu se poate opri executia in timpul efectuarii secventei



Programarea secventiala

Cea mai buna solutie pentru acest VI este de a folosi structura Case cu *error cluster* legat la *case selectors*



Exercitiul

- Calcul Timp ExecVI
  - Aplicatie pentru a utiliza structura “Secventa”

Exercitiul Calcul Timp ExecVI