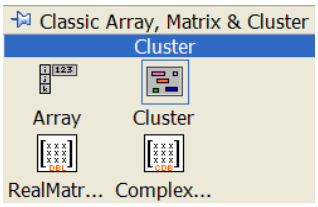
Curs 5 12. 2020

Prof. dr.ing. Iulian Lupea

1. Date de tip structură



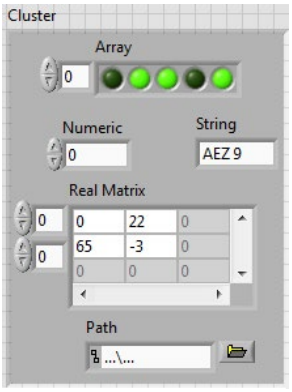
PF: Cluster

1.1. Structura(cluster) este un mănunchi sau o colecţie ordonată formată din unul sau mai multe elemente de tipuri diferite sau de acelaşi tip.

Structura în PF:
paleta *ClasicArray*, *Matrix&Cluster*:
se alege: **indicator** sau **control**

Fiecare element din structură are asociat un *număr de ordine*: 0, 1, 2, 3…

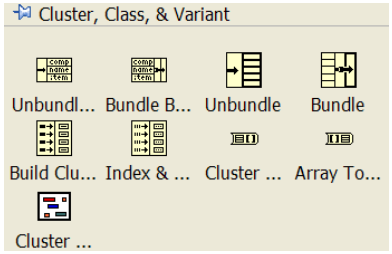
Prin comanda *Reorder Controls in Cluster…* din meniul pop-up asociat cadrului structurii se poate schimba numărul de ordine.



Cluster din 5 elemente

In structura alăturată observăm ***cinci câmpuri*** sau componente:
Array (tablou 1D de valori logice, control)
Numeric (real)
String (şir de caractere)
Real Matrix (matrice reală)
Path (cale spre fişier*): E:\work\test\fisier.txt*

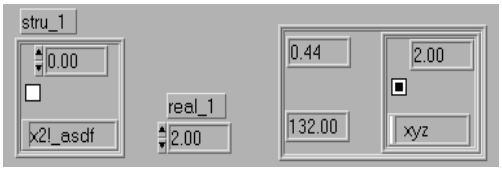
1.2. Schimbarea valorii unei componente, accesul la o componentă
Bundle şi Unbundle

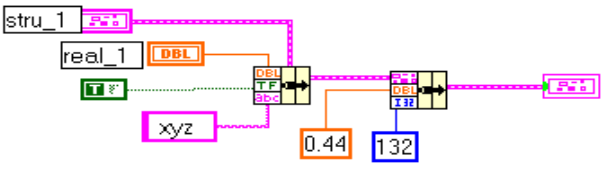


Diagr: Unbundle, bundle, ... , constanta cluster.

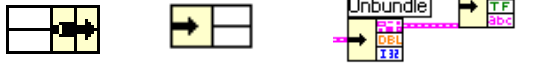
Modificarea valorii unui element al structurii -> se conectează structura la coloana a doua (din mijloc) a operatorului **Bundle**.

Exemplul #1
›se crează un control tip structură cu 3 câmpuri:
numeric, logic, şir caractere
›controlul real_1 (2.0) modifică primul câmp al structurii
›constantele T (logic) şi xyz (şir caractere) modifică următoarele două câmpuri.
›în continuare se formează o nouă structură prin adăugarea unui câmp real şi a unui câmp întreg la structura stru_1.





Pentru a utiliza/ extrage o componentă din structură fără a schimba valoarea ei este folosit operatorul ***Unbundle*** (desface mănunchiul de date).



Funcţii Bundle, Unbundle; unbundle repetat

2.2. ***Bundle By Name şi Unbundle By Name***.

Elementele componente sunt referite prin **numele etichetelor** (nu prin numere de ordine).

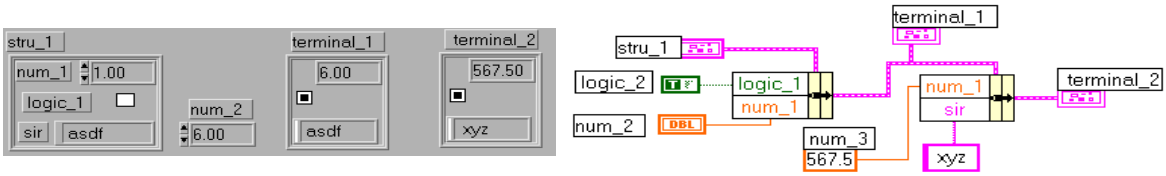
Exemplul #2:

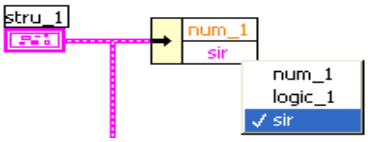
› cele 3 câmpuri ale structurii au etichete (num_1, logic_1, sir)

› două câmpuri ale structurii ***stru_1*** sunt modificate prin ***Bundle By Name***:

logic_1: din F în T;

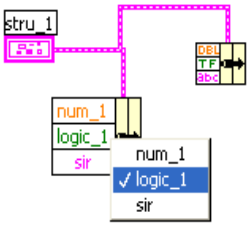
num_1: *din* ***1.0*** *în* ***6.0. etc.***





Acces la 2 câmpuri (num_1, sir) din 3

Unbundle by name



Actualizare valori: Bundle şi Bundle by name

Exemplul 3: Structura unui canal de achiziţie de la senzor tip accelerometru

- *channel info* este numele structurii

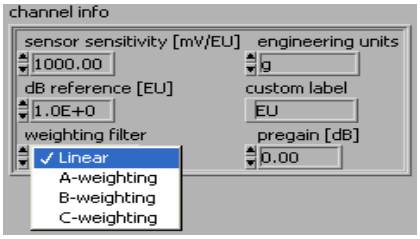


Fig.14.Structură cu şase câmpuri

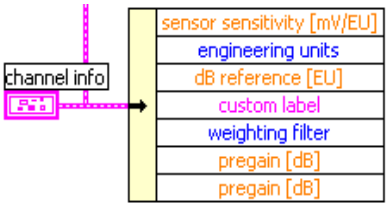


Fig. 15. **Unbundle** by name

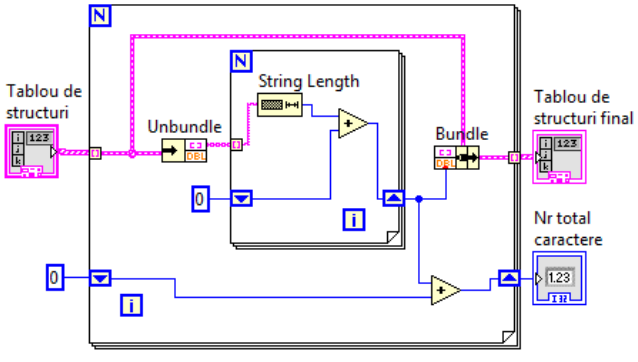
2. Tabloul de structuri

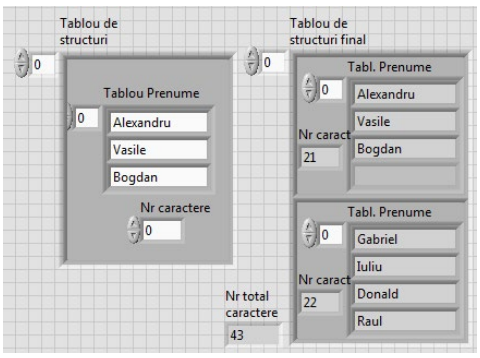
2.1. Se consideră un Tablou de structuri. Structura (cluster) are în compoziţie:

1. un tablou de şiruri de caractere numit *Tablou Prenume* (de exemplu prenumele studentilor dintr-o grupa) şi

2. un câmp numeric numit *Nr. caractere*.

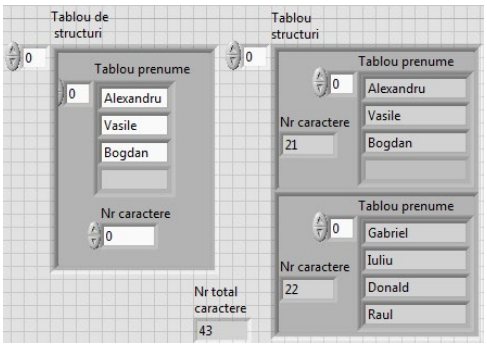
Se ia pe rând fiecare structură din tabloul de structuri şi la fiecare structură se va parcurge tabloul de şiruri de caractere fâcându-se suma caracterelor din toate prenumele din tablou. Se va salva numărul de caractere astfel calculat (din fiecare tablou) în câmpul Nr.caractere din fiecare structură. Se afişează la final tabloul de structuri.

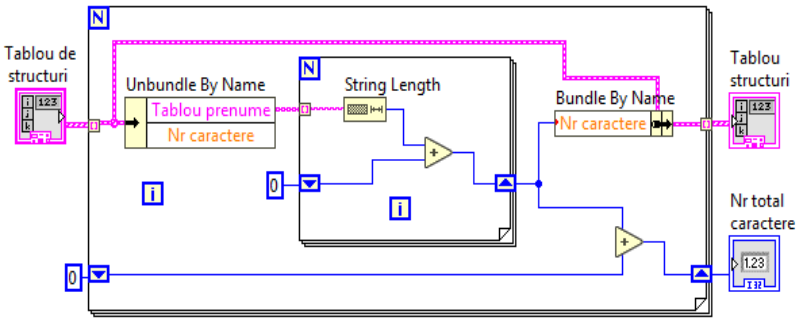




Se parcurge tabloul de structuri cu un ciclu For. Se extrage câmpul Tablou Prenume şi se parcurg elementele tabloului realizând suma numărului de caractere în registrul de transfer. Se extrage numărul de caractere dintr-un prenume cu String Length. La ieşirea din ciclul For interior se salvează valoarea acumulată în registrul de transfer asociat acestui ciclu în câmpul Nr caractere a structurii curente din tabloul de structuri. In paralel folosind un registru shift asociat ciclului exterior se calculează suma totală a caracterelor din întregul tablou de structuri.

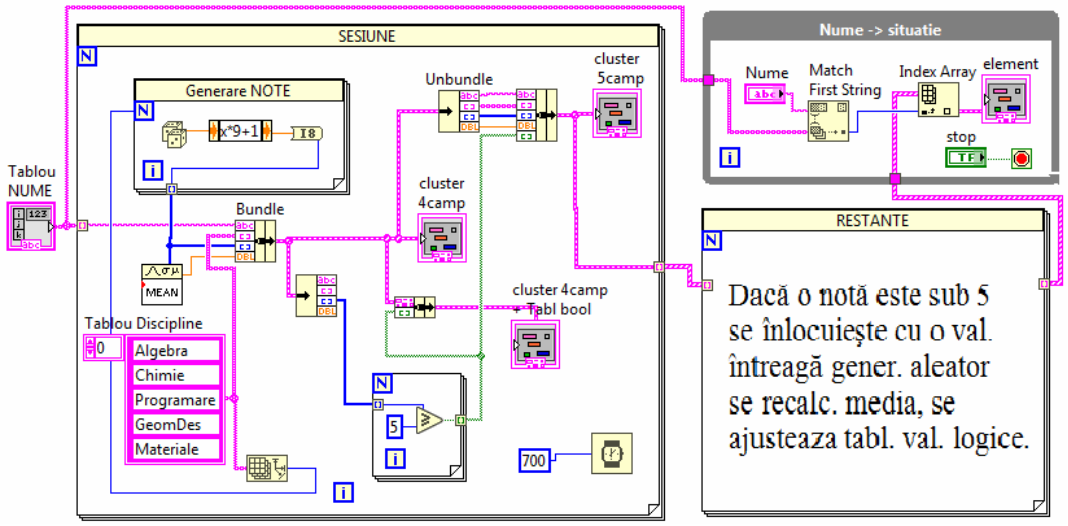
Varianta **Unbundle By Name**





2.2. Aplicaţie cu studenti, discipline, note

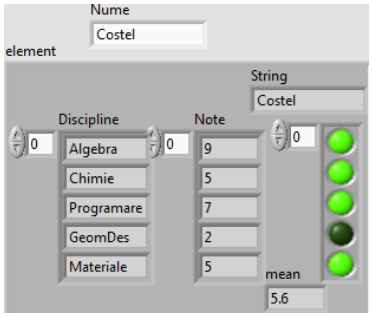
Din tabloul şir de caractere (Tablou NUME) intră câte un nume de student în ciclul For (SESIUNE). Se compune o structură cu 4 câmpuri: şir caractere (nume student), Tablou Discipline, tablou note şi media. In continuare se adaugă un tablou cu 5 val. logice câte una pentru fiecare notă/disciplină. Se generează 'cluster 5camp' o structură cu 5 câmpuri. O altă structură (eticheta 'cluster 4camp+Tabl bool') se afişează şi conţine o structură cu 4 câmpuri şi tabloul cu 5 val. logice.



Tema2: Se completează ciclul RESTANTE conform cu indemnul scris.

In ciclul While (Nume -> situaţie) se selectează şi afişează situaţia studentului introdus în controlul Nume.

Tema3: se foloseşte Bundle by Name / Unbandle by Name în loc de Bundle/ Unbundle.



II. Sir de medii aritmetice

Instrumentul **Mean.vi** calculează media aritmetică a celor n valori din tabloul de intrare X, folosind relaţia (1).

$$media = \frac{1}{n}\sum_{i=1}^{n} X[i] \qquad (1)$$

1.1. Fie un şir de valori: x1, x2, x3, …, xn (reprezentând de exemplu citirile succesive de la un senzor pe un canal de achiziţie) şi se calculează:

M2 media primelor 2 valori,

Mi-1 media primelor i-1 valori,

Mi media primelor i valori …

Mn media tuturor celor n valori.

Vom calcula mai eficient Mi folosind Mi-1 şi valoarea curentă xi.

$\mathbf{M_i = [(i-1)*M_{i-1} + X_i\ ] / i}$ (2)

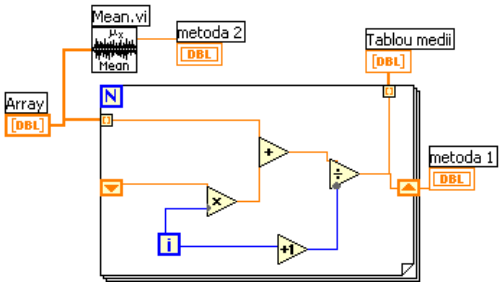
(M0=0)

M1= (0*M0+**X1**) / 1 = X1, iteraţia1

M2= (1*M1+**X2**)/2= (X1+ X2)/2 iteraţia2

M3= (2*M2+**X3**)/3= (X1 + X2 + X3)/3 iteraţia3

…



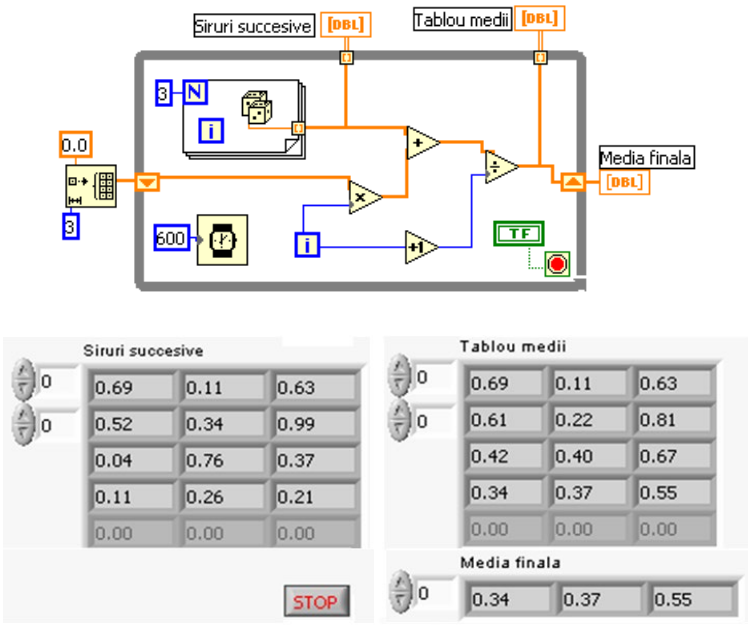
…

Registrul ***transfera media*** la iteratia următoare

Media la fiecare nou element indexat (ciclu) este calculată din media de la iteraţia precedentă păstrată în registrul Shift care se înmulţeşte cu i, rezultând suma primelor i elemente; se adună apoi termenul curent şi se împarte cu (i+1).

1.2. Medieri pe mai multe coloane (individual pe fiecare coloană)

Exemplu: citire de la 3 senzori (prin baleerea repetată a celor 3 canale de achiziţie):

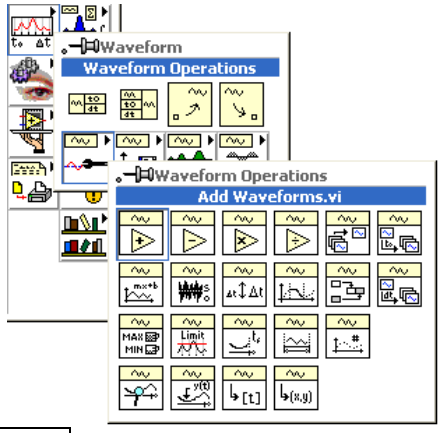


Senzor **#1**↑ **#2**↑ **#3**↑ Medii: #1 #2 #3

Fig. Mediere, independent pe fiecare din cele 3 coloane

*__Tema1__: să se modifice programele pentru calculul **mediei pătratelor** şi **rădăcinii din media pătratelor** pe fiecare din cele 3 coloane.

5. Date de tip <u>formă de undă</u> (Waveform) + salvarea lor în fişiere



5.1. Forma de undă este o **<u>structură</u>** compusă din trei componente sau câmpuri:

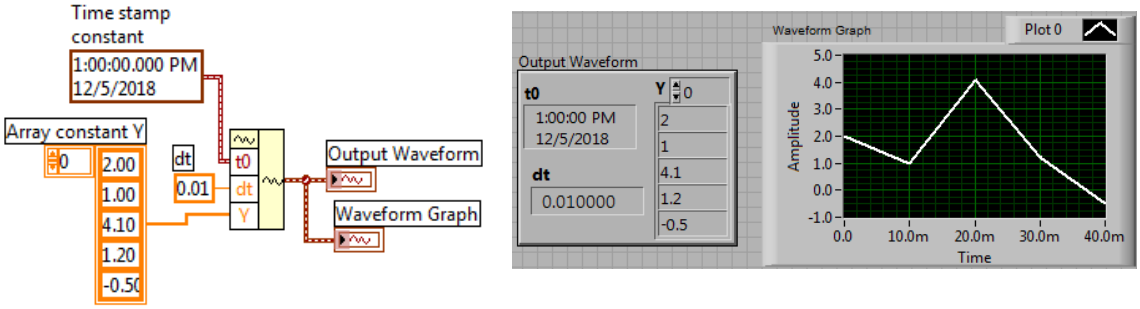
t0 = originea timpului (tip dată: ***time stamp***),
dt = spaţierea dintre două eşantioane succesive şi
Y = tabloul eşantioanelor (valori numerice).

elementul y0 a fost preluat la momentul t0,
y1 la momentul t0+dt,
y2 la t0+2*dt etc.

Momente echidistante: t_0, **Δt**	t_0, t_1, ... , t_{n-1}.
Sir de valori reale asociat: **Y**	y_0, y_1, ... , y_{n-1}

5.2. Funcţia Build Waveform

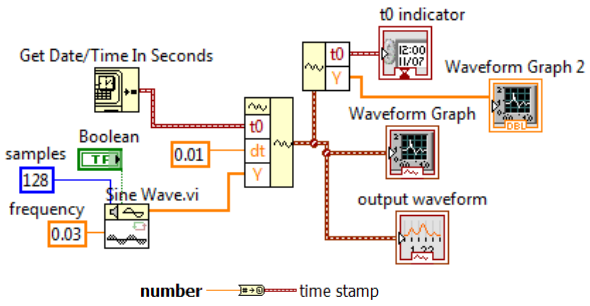
-generează o **formă de undă** folosind **trei constante** în diagramă*:*

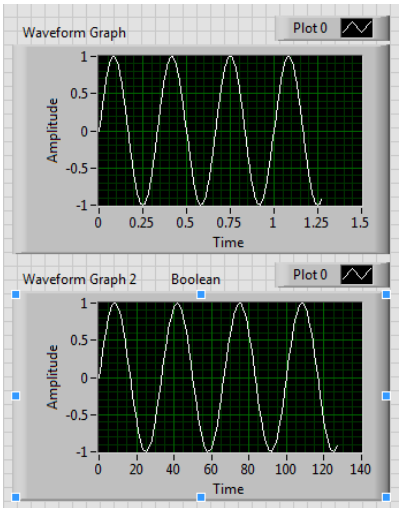


Constanta timp curent, dt , constanta tablou Y
Build waveform *Momente de timp în milisecunde:* 10m 20m ...

5.3. Funcţia ***Get Waveform Components***

Se citeşte **timpul curent** + spaţiere 0.01s + semnal sinus → **Build Waveform**





Se afişează forma de undă generată (grafic şi valori) + ***Get Waveform Components*** -> tablou Y -> grafic.

Observăm:
*** spaţierea fireasca => 128*0.01=1.28 secunde**
durata formei de undă
***spaţiere 1 la reprezentarea tabloului eşantioanelor generate de *sine wave.vi*.**

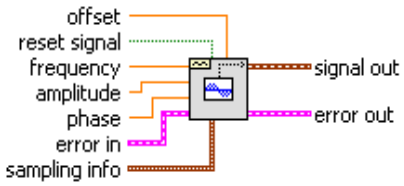
5.4. Prezentare ***Sine Waveform.vi***

- intrari (rata de eş. şi frecvenţa) raportate pe secundă

Structura ***sampling info*** conţine:

1) **Fs= 1000** (implicit) -> rata de eşantionare (eşantioane pe secundă) şi

2) **#eş=1000** (implicit) -> numărul total de eşantioane din waveform.



Sine waveform.vi

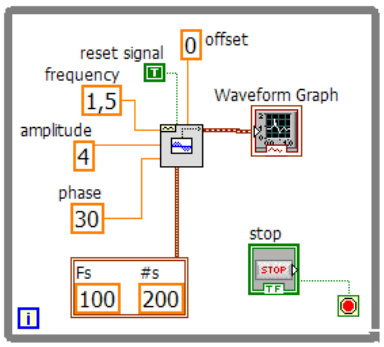
frequency= număr cicluri pe secunda [**Hz**] (observăm 1,5 perioade/1s => 3per./2sec

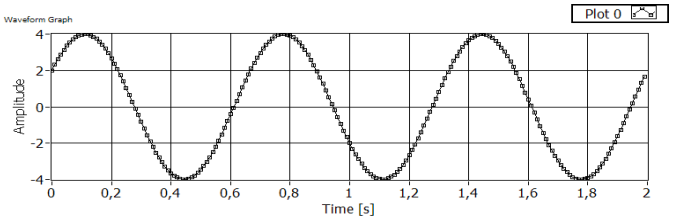
iar semnalul durează 2s)

offset=translaţie pe oy

phase [grade]

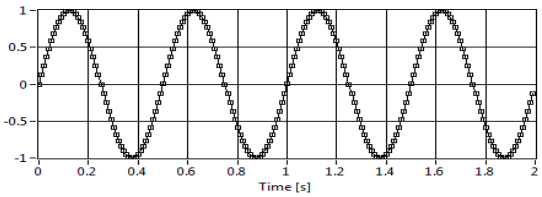
= faza este considerată dacă **reset signal**=T



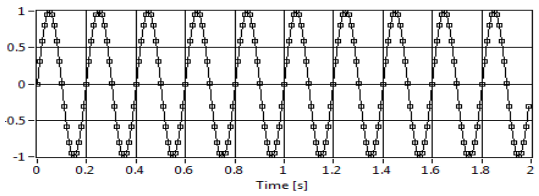


Exemplu: **Fs=100eş/s, #eş=200 =>durata=2sec, frequency=1,5 Hz**

Alte frecvenţe:

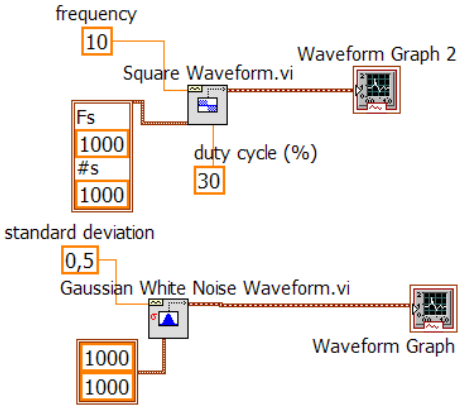


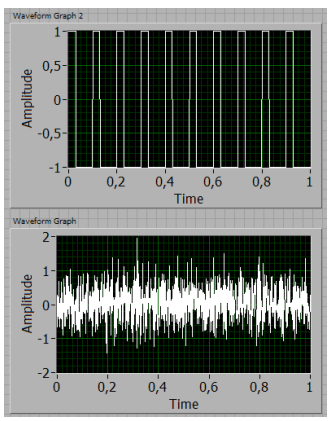
frequency=2 Hz,

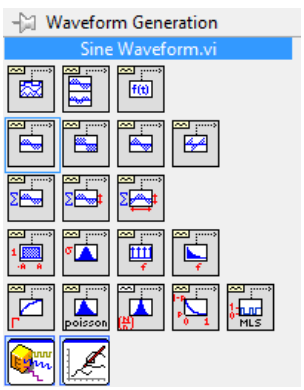


frequency=5 Hz; faza=0

5.3. **Square Waveform**.vi şi **Gaussian White Noise Waveform**.vi:

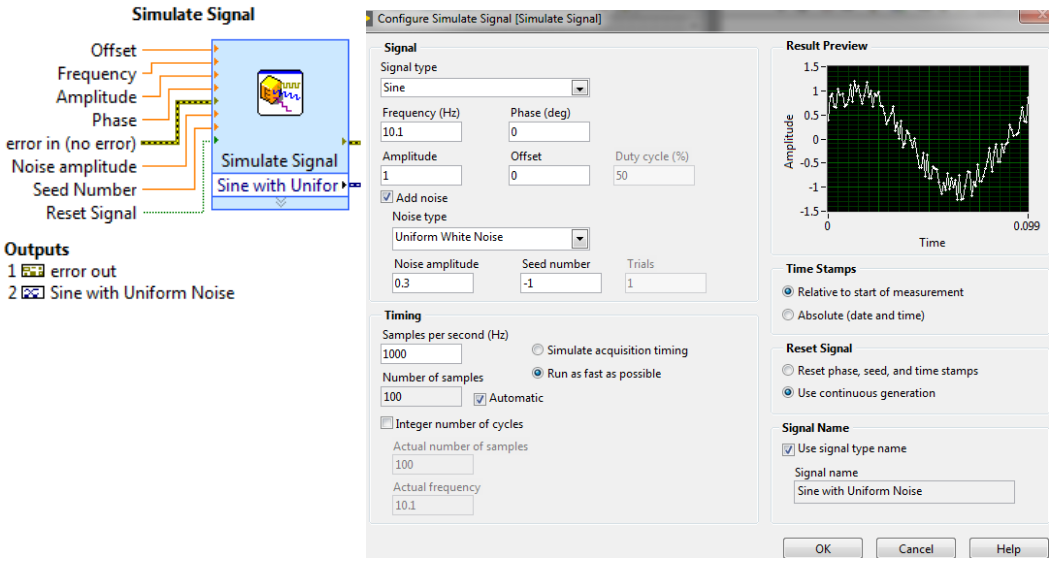




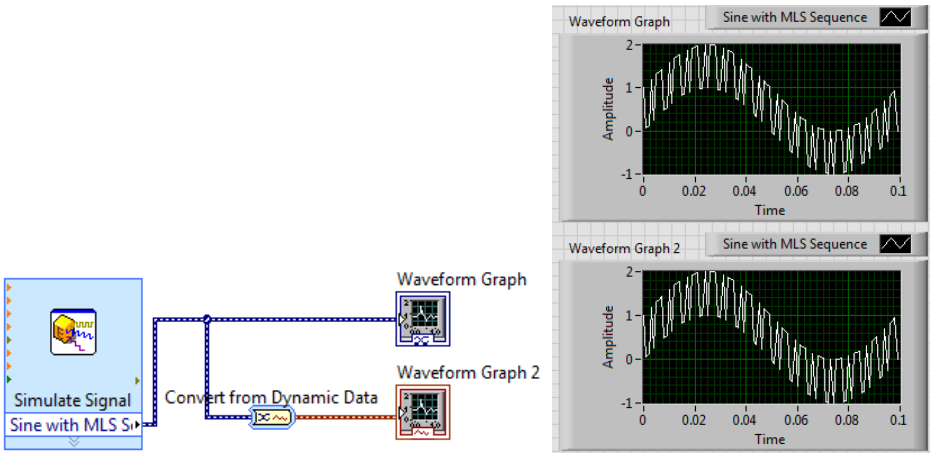


5.4. Funcţia Simulate Signal - Express VI

Un Express VI este un instrument virtual a cărui is setări se pot face interactiv în cadrul unei cutii de dialog.
Simulează a sine wave, square wave, triangle wave, sawtooth wave, or noise signal
Express VI returnează în general semnal ieşire=**Dynamic Data**

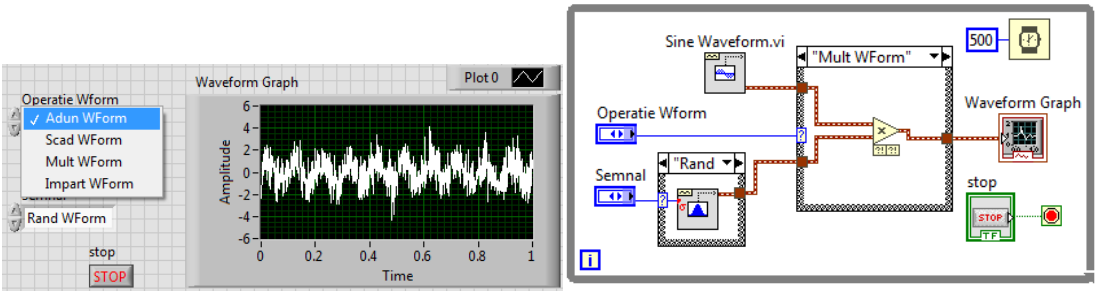


Funcţia Convert from dynamic data din paleta Express/ Signal Manipulations
Conversie din Dynamic data în Single waveform sau 1D array de waveforms sau în

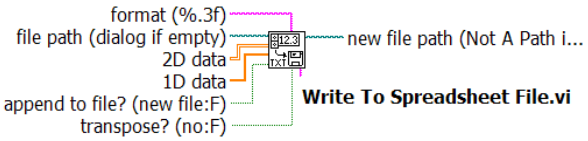


5.4. Operaţii cu forme de undă

Două forme de undă se pot aduna, scădea, înmulţi şi împărţi; cele două trebuie să aiba aceeasi spaţiere **dt** între eşantioane**.**



5.4. Salvare tablouri numerice în fişier



Write To Spreadsheet File.vi

-realizează salvarea unui **tablou** de valori reale **1D sau 2D**, în fişier text (textul poate fi citit cu Wordpad)

- fişierele create *lisa1* şi *lisa2* (în final) vor avea **acelaşi conţinut**
- ***append to file=T*** (evita suprascrierea)

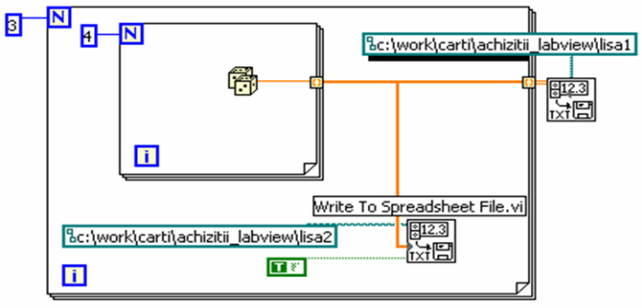
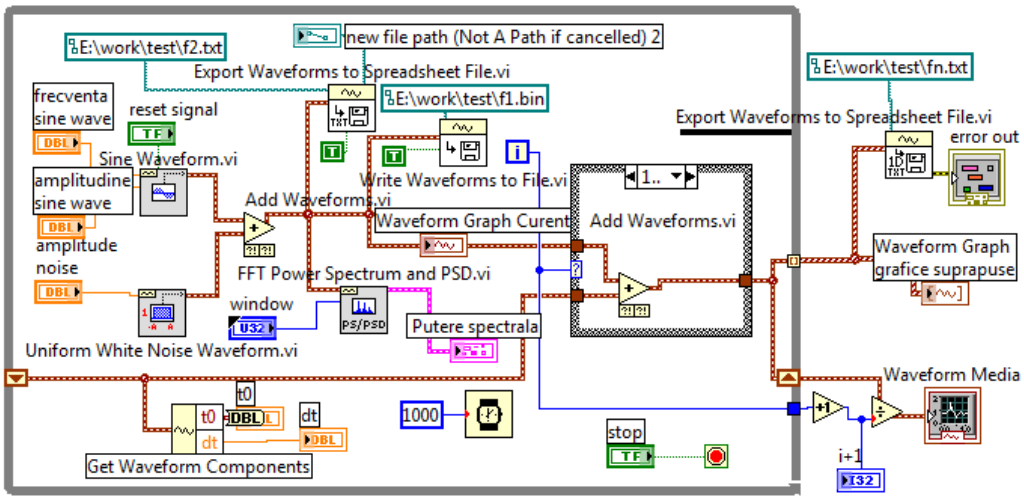


Fig. 3

5.5. Mediere Sine Waveform + zgomot şi salvare forme de undă în fişier



La tunelul de ieşire din ciclu avem **1D array of waveform** (tablou de forme de undă)

Sine Waveform.vi şi Uniform White Noise Waveform.vi sunt plasate în subpaleta Waveform Generation.

1) **Export** Waveforms to Spreadsheet File.vi → salvează în format **text** forma de undă

```
Waveform    [0]    [1]    [2]    [3]
t0    1/1/1904  03:00:00.000000     1/1/1904  03:00:01.000000     1/1/1904  03:00:02.000000
delta t     0.001000    0.001000    0.001000    0.001000

time  Y[0]  Y[1]  Y[2]  Y[3]
1/1/1904  03:00:00.000000     1.170654E-1 1.103271E-1 1.626221E-1 7.844238E-2
1/1/1904  03:00:00.001000     2.188086E-1 5.486205E-2 8.740837E-3 1.155914E-2
1/1/1904  03:00:00.002000     3.132239E-1 1.694494E-1 2.224524E-1 1.533484E-1
1/1/1904  03:00:00.003000     3.555332E-1 2.795871E-1 1.281131E-1 2.006427E-1
1/1/1904  03:00:00.004000     2.628989E-1 9.076752E-2 2.347434E-1 2.999075E-1
1/1/1904  03:00:00.005000     4.894003E-1 2.610495E-1 3.725454E-1 4.442968E-1
1/1/1904  03:00:00.006000     5.251802E-1 5.397310E-1 2.647432E-1 2.460237E-1
```

Fig.20 fişier **fn.txt**; primele 3 linii: 4 valori t0, 4 valori delta t =0.001000,
5 coloane: timp, Y[0], Y[1], Y[2], Y[3]

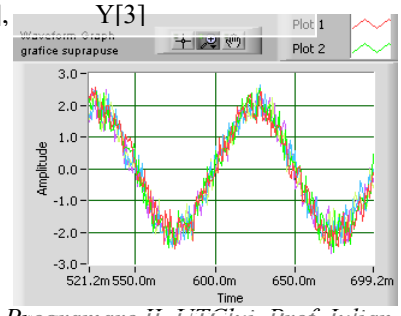
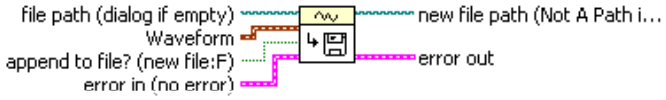


Fig. 19. Grafice suprapuse

2) Salvare în format **binar** (Write Wavefom to File.vi)



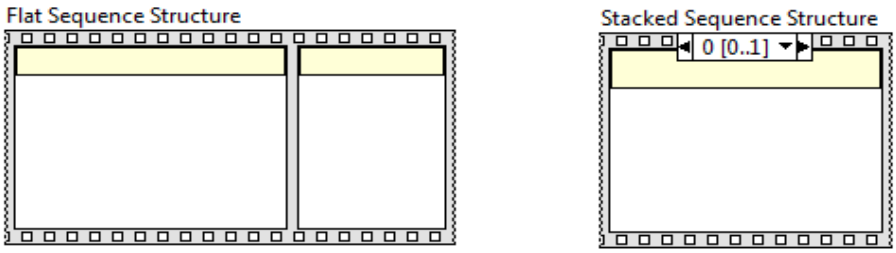
3) ***Write** Waveforms to File.vi* scrie în format **binar**

6. Structura secvenţială (continuare)

6.1. Flat Sequence Structure versus Stacked Sequence Structure

-una sau mai multe **subdiagrame** numerotate (0, 1, 2 ...) care sunt **executate secvenţial**, în ordine crescătoare,

-comenzi: Add Frame After, Add Frame Before,
Duplicate Frame, Delete This Frame,
Show Frame, schimbarea ordinii: Make This Frame {0,1,…},



1. informaţia primită din exterior la un tunel de intrare, este disponibilă (se poate accesa) în oricare dintre secvenţe,
2. prin **terminale secvenţiale locale** (***Add Sequence Local***) se pot trece date calculate într-o subdiagramă (săgeată spre exterior) spre **toate** subdiagramele următoare (săgeată spre interior la secv. următoare),
3. valorile calculate conectate la **tuneluri de ieşire** vor fi disponibile la exterior **numai la terminarea execuţiei tuturor secvenţelor structurii**.

6.2. Aplicaţie *Add Sequence Local*

Constanta 0.0 de la tunelul de intrare este folosită numai în secvenţa a doua (indice 1) pentru însumarea elem. din tabloul crescut.
Tablou (1D) de la tunelul de intrare este folosit numai în prima secvenţă (indice 0)
Tunelurile de ieşire spre Suma şi Tablou+1 sunt parcurse după terminarea celor două secvenţe.

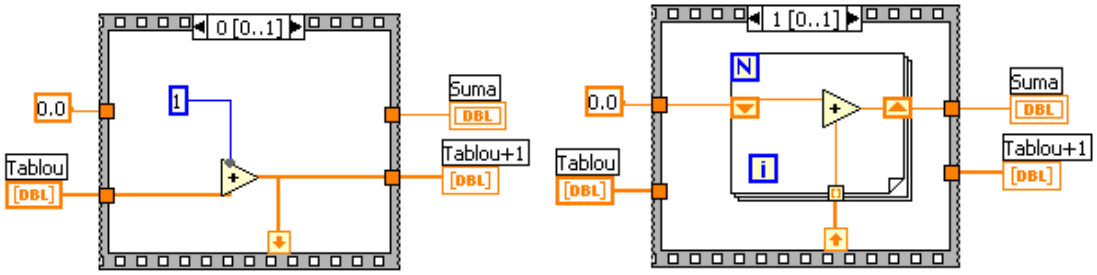
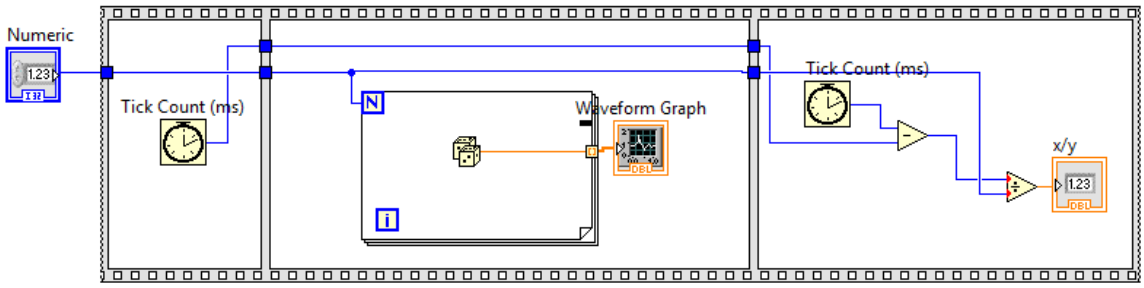


Fig. 8 Secv.0 şi 1 se execută în ordine; Tablou+1 trece in secv.1 unde se însum. elementele

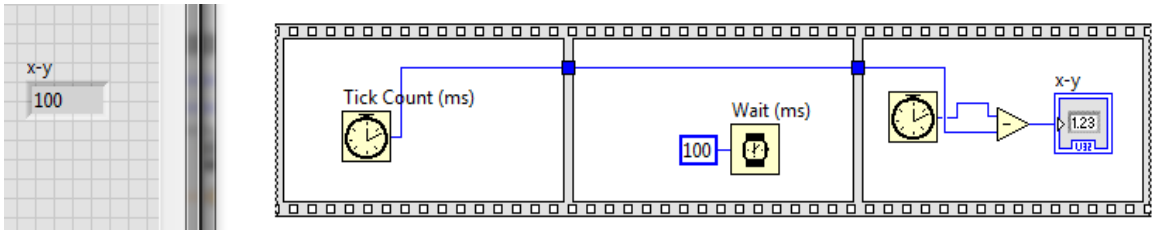
2. Măsurare timp consumat /Time Passed la execuţia unui VI

Se propune: funcţia **Tick Count (ms)** + structura Sequence

Ex. Se măsoară timpul necesar unei ciclări:



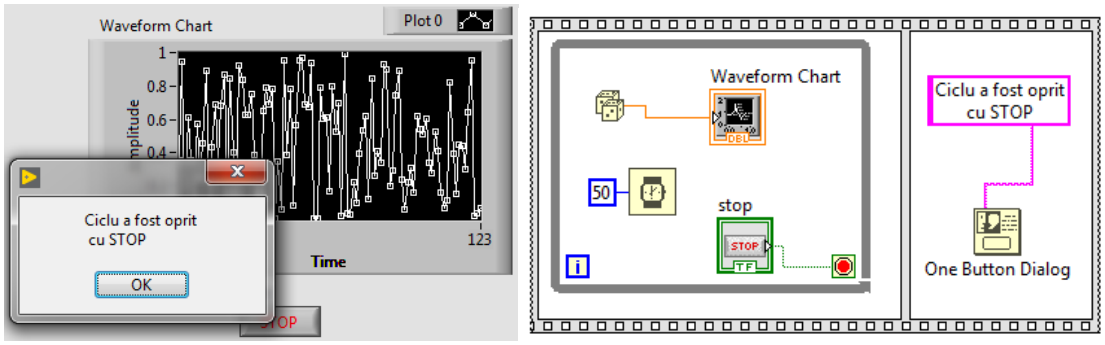
Ex. Se verifică egalitatea dintre timpul măsurat şi timpul de aşteptare Wait (ms):



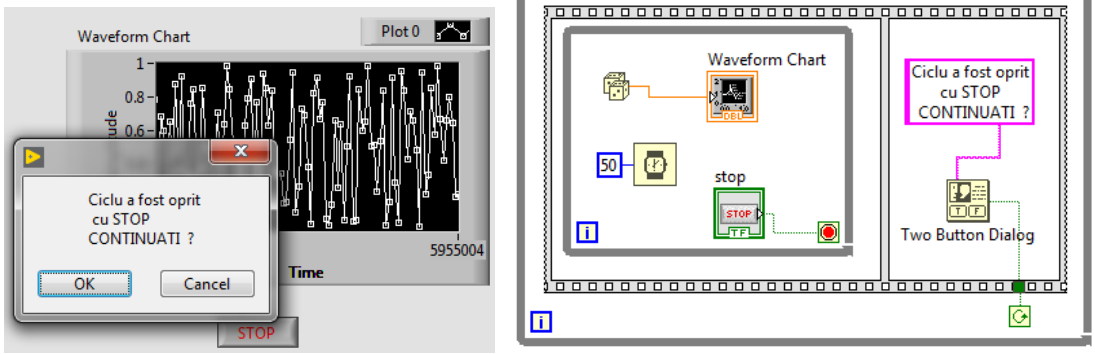
Obs: valoarea în msec a timer-ului trece de la (2^32)–1 la→ 0.

Exemplul 3

Varianta 1: la apăsare buton **stop** se opreşte ciclul + **fereastră cu mesaj**



Varianta 2: apăsare stop → oprire ciclul interior → continuare /oprire ciclu exterior



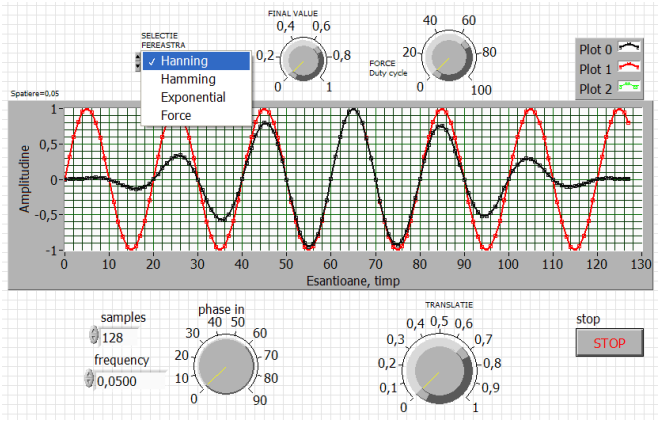
VIII. Prelucrare tablou numeric prin funcţii fereastră (windowing)

- cod C şi Matlab în digrama Labview

Se recomandă prelucrarea prin ferestre de ponderare a semnalului achiziţionat înainte de aplicarea transformatei Fourier semnalului, cu scopul de a minimiza apariţia de frecvenţe false inexistente în semnalul achiziţionat.

In aplicaţia curentă se generează un semnal sinusoidal discret apelând **Sine Wave.vi** (samples=***n***, frequency, phase in). Semnalul poate fi prelucrat prin una din cele patru ferestre de ponderare predefinite (Signal Processing/ Windows):

Hanning, Hamming, Exponenţial şi Forţă.



Fiecare eşantion din semnal este înmulţit (ponderat) cu câte un coeficient specific ferestrei (total n înmulţiri); coeficienţii unei ferestre sunt calculaţi prin relaţii de calcul specifice ferestrei.

In aplicaţie, semnalul este prelucrat automat de o fereastră selectată de operator şi în paralel acelaşi semnal este înmulţit (prin program) cu coeficienţi generaţi după relaţia de calcul, rezultând două semnale identice suprapuse.

In figura 1 observăm curba sinus iniţială de amplitudine constantă şi două curbe identice suprapuse, prelucrate de fereastra Hanning (cu amplitudine zero la început şi la sfârşit) şi respectiv curba obţinută separat prin formula de calcul →

$$y_i = 0.5x_i[1 - \cos(w)]$$

$$w = \frac{2\pi i}{n} \qquad i = 0,\ 1,\ 2,\ \ldots,\ n-1,$$

Formule calcul Fer. Hanning

Indice eşantion i=	0	1	2	…	60	61	…	126	127
Semnal sinus **iniţial x_i** = Phase in=90grade (cos)	1	0.951	0.809	…	1	0.951	…	-0,309	-0,588
Relaţii de calcul cu *Fereastra **Hanning:***	Cod **Matlab**: i= 0 : n-1; w=2*pi*i/n; **y**=0.5***x** .*(1-cos(w));								
Semnal **rezultat** y_i=	0,00	0,001	0,002	…	0.99	0.946	…	-0.001	0,000

Ponderarea cu coeficienţii ferestrei Hanning se face într-un caz separat printr-un segment de cod scris în Matlab (Fig. 3).

x_i sunt eşantioanele semnalului sinus iniţial iar

y_i sunt valorile semnalului rezultat.

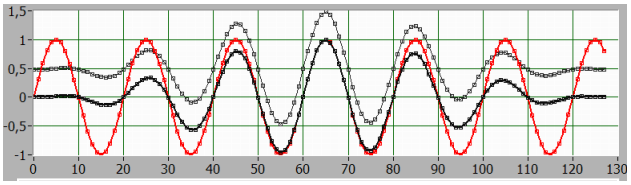
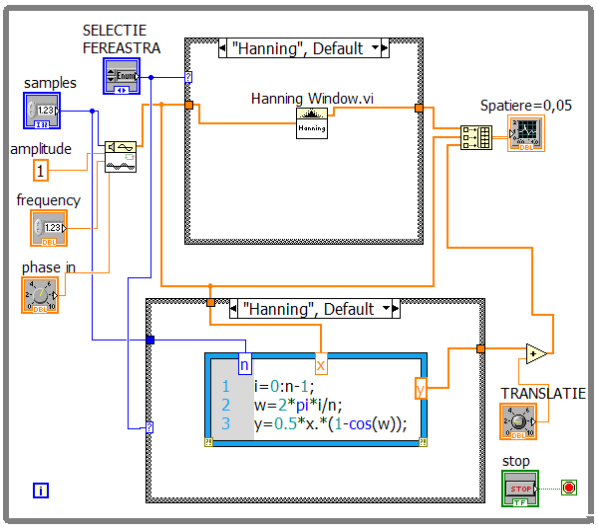
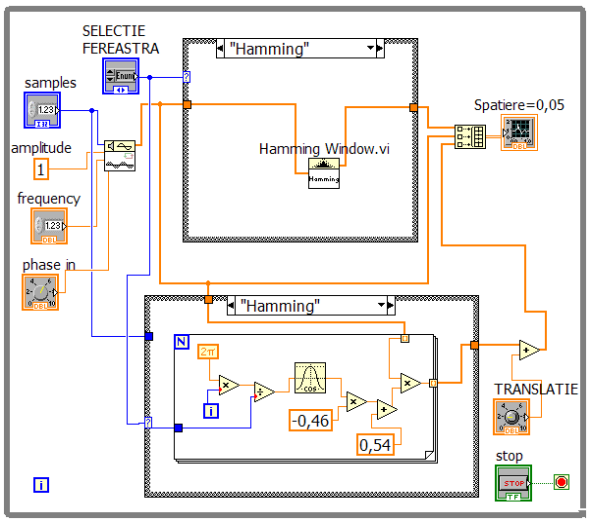


Fig. Actiune Fereastră Hanning

Controlul TRANSLATIE permite separarea celor două curbe identice (Fig. 2). Prin aceasta se verifică corectitudinea programării explicite asociate ferestrei predefinite.

> Sunt folosite două instrucţiuni Case (4 cazuri în fiecare) şi un selector comun: Selectie Fereastra .





> Build Array uneşte 3 semnale 1D într-unul 2D pentru afişare într-o singură fereastră grafică.

Ponderarea cu fereastra Hamming se face în paralel într-un alt caz folosind programare Labview. y_0 şi y_{n-1} sunt val. minime.

$$y_i = x_i[0.54 - 0.46\cos(w)]$$

$$w = \frac{2\pi i}{n} \quad i = 0,\ 1,\ 2,\ \ldots,\ n-1$$

Calcul Fer. **Hamming**

$$y_i = x_i \exp(a*i)$$

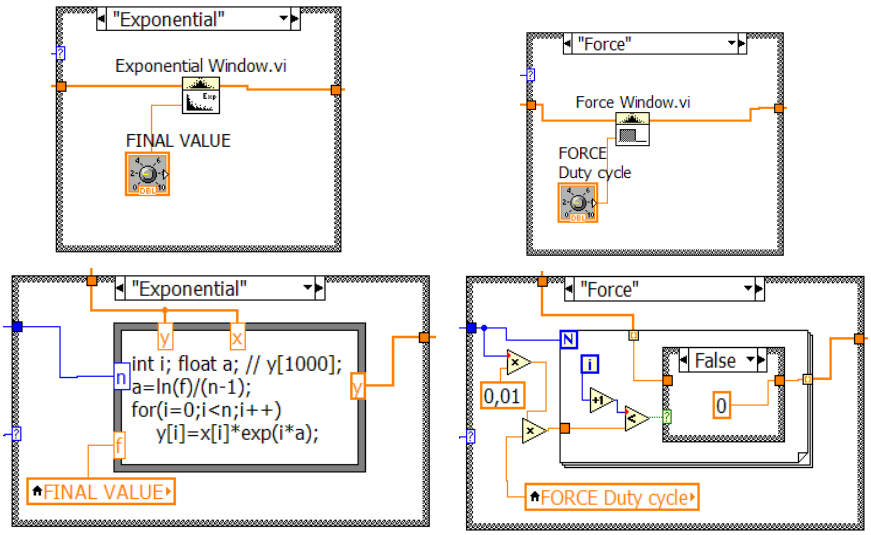
$$a = \frac{\ln(f)}{n-1} \quad i = 0,\ 1,\ 2,\ \ldots,\ n-1$$

Calcul Fer. **Exponenţial**

$$w = \begin{cases} x_i & (\text{if } 0 \le i \le d) \\ 0 & \text{elsewhere} \end{cases}$$

$$d = (0.01)(n)(\textbf{duty cycle})$$

Calcul Fer. **Force**



• La ponderarea cu fer. Exponentială se apelează la formula nod folosind codul limbajului C.
Final value (f) unde **$0 \le f \le 1$,** este valoarea finală minimă, controlată din panoul frontal (pentru f >1 se obţine creştere în locul diminuării semnalului în timp)

• La ponderarea cu fereastra Force se apelează la programare în Labview.
Controlul **Duty cycle %** este în procente între 0% şi 100%.
Dacă duty cycle =100 rezultă d=n, unde n = numărul de elemente din X iar yi=xi pentru $0 \le i \le d$

TEME:
1. Modificaţi controalele Final Value şi Duty cycle asociate ferestrelor Exponential şi Force observând efectul asupra semnalului ,

2. Adăugaţi în aplicaţie şi semnalul Square wave.vi pe lângă Sine wave.vi si selectaţi semnalul dorit.
3. Adăugaţi fereastra Blackman (cazul 5),
4.Folosiţi o singură instrucţiune CASE în aplicaţie unificând cazurile omoloage.

Obs: *efect Windowing→diminuare spectral leakage*

Fereastra Hanning anulează prima valoare din semnal dar Hamming nu (modificaţi *faze in* pentru a porni semnalul de la o valoare nenulă şi a vedea comparativ ferestrele).

Efectul ferestrei **Exponential** asupra unui semnal sinusoidal:

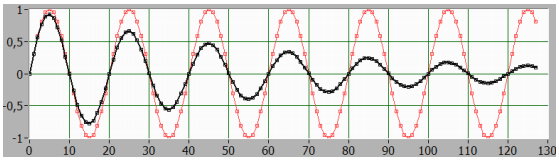


Fig. Final value =0.12 => diminuare medie

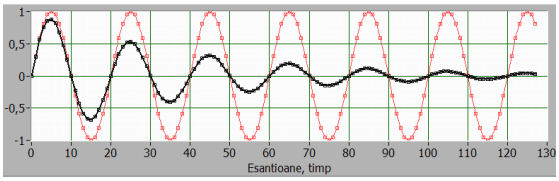


Fig. Final value =0.04 => diminuare mare

Efectul ferestrei **Force** asupra unui semnal sinusoidal:

65% semnal neafectat:

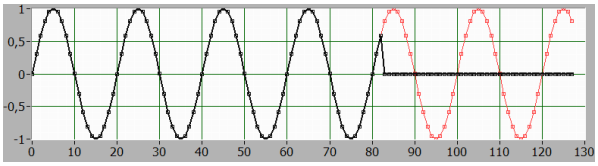


Fig. Force **duty cycle= 65%,** phase in=0

35% semnal neafectat:

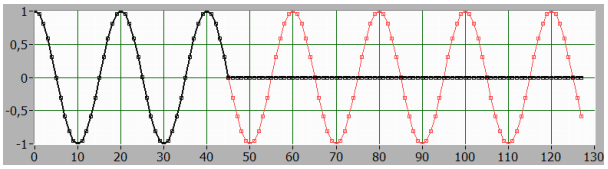


Fig. Force **duty cycle= 35%,** phase in=90

Efect fer. Hanning asupra semnalului UWN:

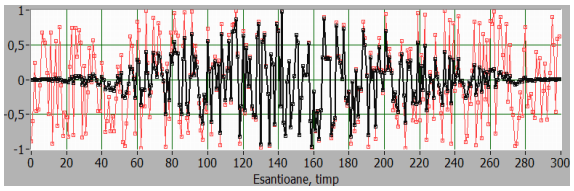


Fig. Uniform White Noise (roşu) + Hanning

Efect fer. Exponential asupra UWN:

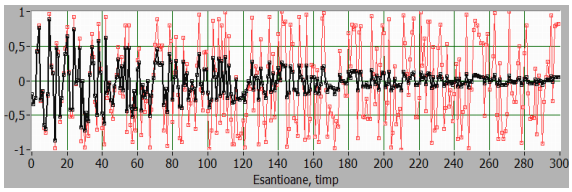


Fig. UWN + Exponential

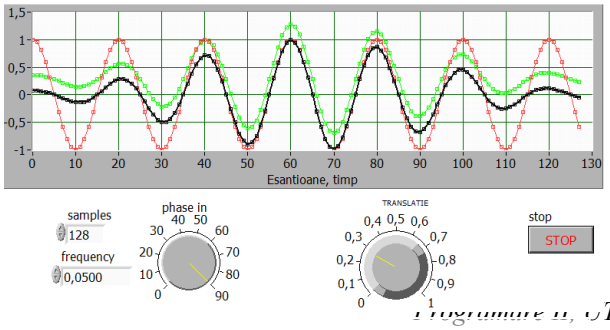


Fig. Hamming (nu începe şi nu se termină la 0)

Programare pe EVENIMENTE, partea 1 - (Event-Driven Programming)

1. Fluxul datelor în diagramă

* utilă în special pentru dezvoltarea interfeţelor utilizator (UI)

- tradiţional este folosită tehnica ***polling*** *=citirea repetată a controalelor din PF pentru a sesiza o modificare*

ciclarea deasă → utilizează resurse mari CPU

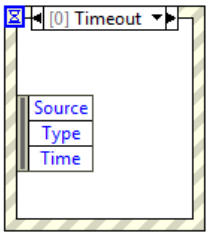
ciclarea mai rară → permite scăparea unor evenimente sau a ordinii apariţiei lor.

1.1. Fluxul firesc al datelor în diagramă:

un nod al diagramei se execută când are la dispoziţie date la toate intrările conectate. După execuţie nodul generează date care trec prin fire spre nodurile următoare conectate conform cu fluxul de date.

1.2. Programarea condusă de evenimente permite modificarea execuţiei fireşti la apariţia unui eveniment urmată de tratarea acelui eveniment. Evenimente pot fi: schimbarea valorii unui control, intrare sau ieşire mouse intr-o/dintr-o zona de pe PF, apăsare buton mouse, inchiderea sau schimbarea mărimii unei ferestre, apăsarea unei taste, timeout etc

- programul aşteaptă apariţia unui eveniment (deplasare mouse, apăsare tastă etc), tratează evenimentul apărut şi revine la aşteptare
- se foloseşte ***structura eveniment*** (Event Structure)



2. **Descrierea structurii Eveniment** (Event Structure)

Structura Eveniment poate avea una sau mai multe subdiagrame sau cazuri.

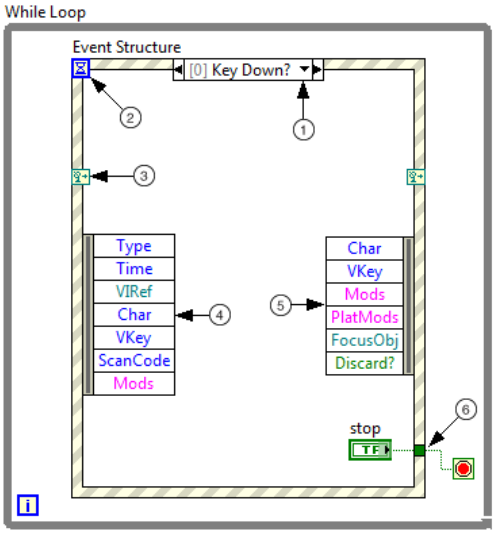
* fiecare subdiagramă/caz gestionează unul sau mai multe evenimente

Cât timp se aşteaptă apariţia unui eveniment nu se consumă resurse de calcul CPU.

Evenimentele se pun în **coada de aşteptare** dacă nu pot fi tratate imediat (de exemplu un eveniment apare când un alt eveniment se execută).

Se pot şterge elemente din coada , folosind funcţia Flush Event Queue.

Evenimentele sunt statice sau dinamice.



Exemplu de Event structure cu eveniment tip Key Down? event case

1. selector de evenimente + Eticheta sau antetul
   - descrie tipul evenimentelor pentru acel caz.

2. Terminalul Timeout
   specifică numărul de milisecunde de aşteptare după care se execută cazul Timeout (dacă -1 => aşteptare nelimitată)
   - dacă se conectează o valoare în msec trebuie să existe şi un case Timeout altfel eroare.

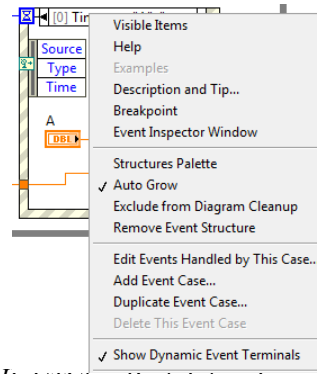
3. Setare Terminal pentru eveniment dinamic

-bifează Show Dynamic Event Terminal la care se leagă ieşirea 1 a funcţiei Register For Events astfel încât la crearea unui nou caz să apară în fereastra de editare posibilitatea de a creea un Eveniment dinamic.

1. o referinţă pt. înregistrare Eveniment (event registration refnum)
2. un cluster de referinţe la evenimente.

Se poate lega şi terminalul interior dreapta caz în care terminalul nu va mai avea aceeaşi valoare ca şi terminalul stâng. Se poate lega/wire la terminalul stâng:

1.referinţa refnum de înregistrare eveniment sau

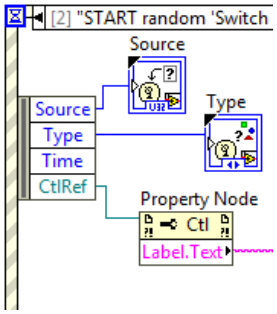


2.cluster de referinţe de înregistrare evenimente

4. Nodul **Event Data** (plasat interior contur stânga)

- permite acces la datele **pe care LabVIEW le returnează la apariţia unui eveniment (date despre eveniment)**.
- similar cu **Unbundle By Name** se poate redimensiona numărul de terminale din nod pe verticală şi se poate selecta articolul dorit.
- nodul Event Data permite acces la elemente cum sunt:

Type and **Time**, elemente comune la toate evenimentele
Char and **VKey** elem. specifice evenimentului apărut.



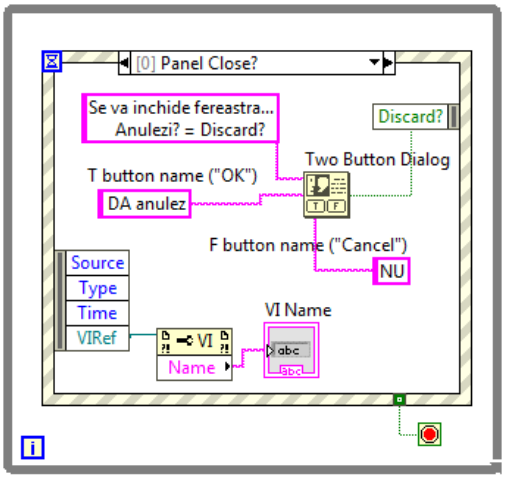
5. Nod **Event Filter** identifică date ale evenimentului pe care le poţi modifica înainte ca (UI) interfaţa utilizator să poată prelucra acea dată. Nodul apare numai în cazuri de tip filter events (cu ? la urmă). Dacă se doreşte schimbarea **event data**, se poate lega la dreapta şi modifica articolul cu data în cauză sau lega data din Event Data Node la Event Filter Node. Astfel se pot schimba date ale evenimentului cu noi valori =cele dorite.

De exemplu pentru eliminarea/anularea unui eveniment deja cerut se leagă valoarea True la terminalul **Discard?**

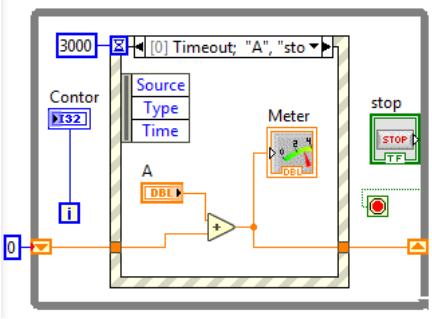
Dacă nu se conectează o valoare la un terminal de date aparţinând nodului Event Filter acea dată îşi păstrează valoarea

Alt exemplu:

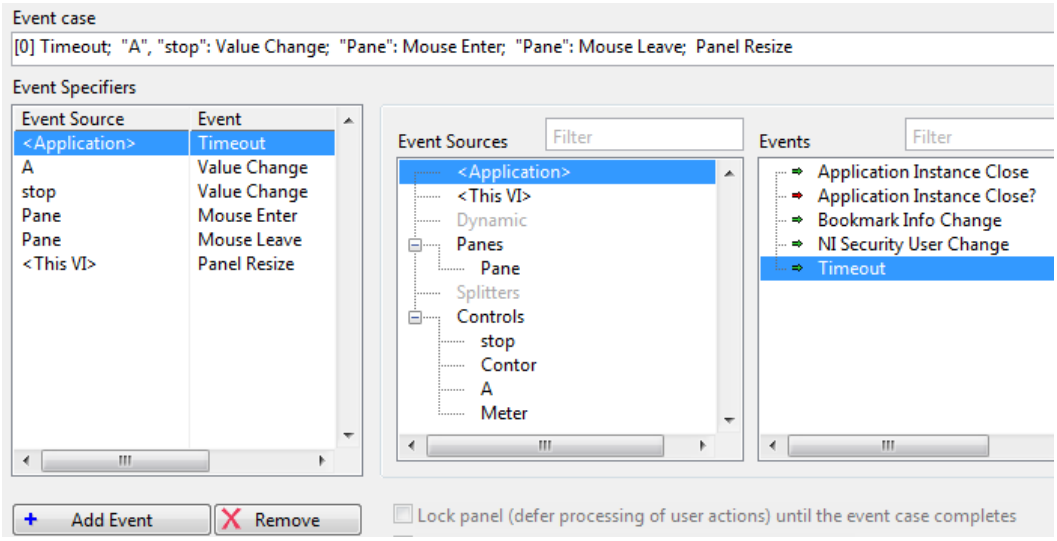
"Key down" versus "Key down?"

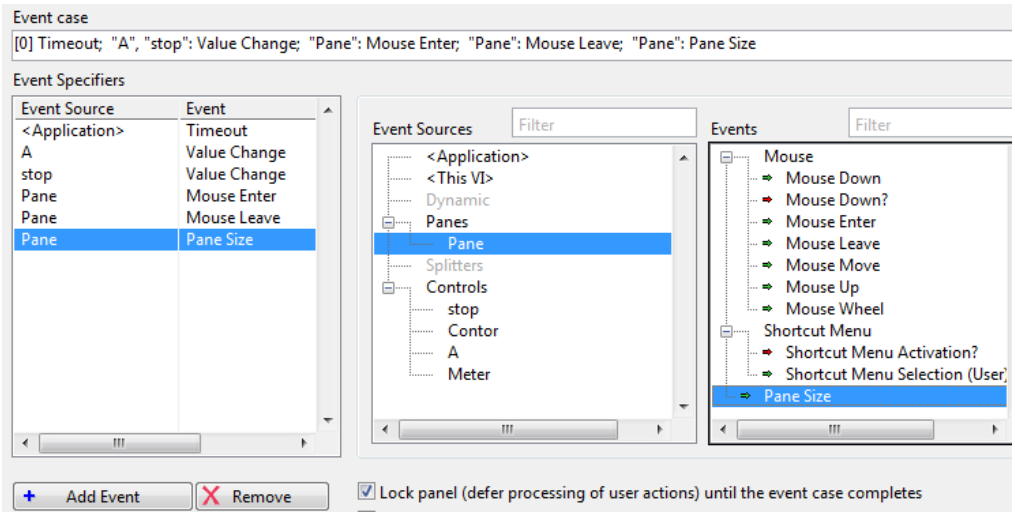


6. Event structure suportă tuneluri de ieşire dar nu trebuie legături la tunel din fiecare caz ca în Structura Case. La tunelurile nelegate se foloseşte valoarea implicită pentru tipul datei tunelului (Right-click a tunnel and deselect **Use Default If Unwired** from the shortcut menu to revert to the default Case structure behavior where tunnels must be wired in all cases. You also can configure the tunnels to wire the input and output tunnels automatically in unwired cases).



Fereastra de construire a unui EVENT CASE



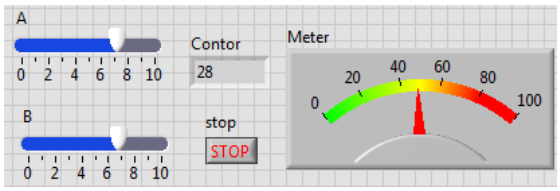


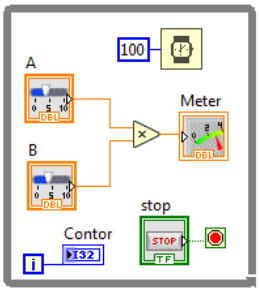
3. Exemple

3.1. PF interactiv - Ciclul While (polling) versus Event Structure

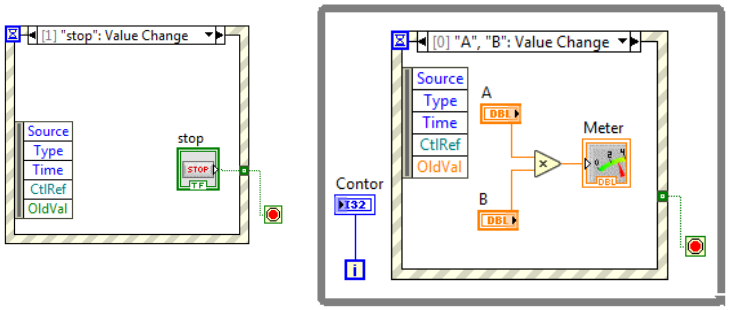
Efectuăm produsul a două controale numerice A şi B cu observare permanentă a modificării valorilor şi a produsului valorilor.

1.1. Varianta#1: produsul plasat în corpul ciclului While. Când nu se modifică valorile A sau B ciclarea continuă consumând resurse de calcul.

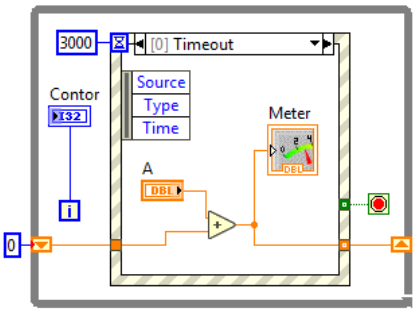




1.2. Varianta#2: programare pe evenimente. Ciclul while se păstrează dar produsul este plasat în primul caz *'[0]"A","B":Value Change'* al structurii Event, creat prin modificarea cazului '0 Timeout' cu comanda 'Edit Events Handled by This Case'. La execuţie se observă că ciclarea este suspendată (Contorul nu creşte) când nu sunt înregistrate evenimente de tipul modificării valorii controalelor A sau B. Când schimbăm din PF valorile controalelor A sau B contorul ciclului creşte. Oprirea ciclului şi a aplicaţiei este posibilă numai dacă se introduce un al doilea caz de tip Value Change în structura Event care conţine terminalul butonului de STOP. Acest caz se execută la apăsarea STOP în PF iar la ieşirea din caz (după tratare Event), valoarea True a butonului opreşte ciclul While. Obs. se pot trata cele trei evenimente într-un singur caz.



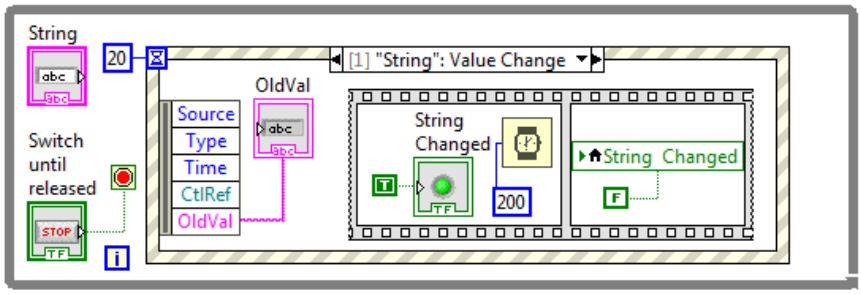
1.3. **Cazul Timeout** al structurii Event. Aplicaţia conţine cazul Timeout şi cazul tip Value Change pentru butonul de STOP al ciclului. La fiecare 3000ms de aşteptare daca nu apare un eveniment se execută automat cazul 'Timeout' in care se adună repetitiv valoarea lui A. Dacă schimbăm val. A din PF nu se întâmplă nimic imediat (doar dupa 3000ms se aduna valoarea schimbată a lui A) fiindcă aceasta nu este tratată ca eveniment.

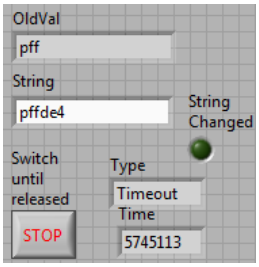


Tema1: adăugaţi un caz nou pentru sesizarea schimbării valorii lui A şi actualizarea imediată a sumei (vezi şi variabile locale). Tema2: Includeţi în cazul Timeout şi evenimentul 'schimbare valoare A' (Value Chance).

3.2. Eveniment static (acţiune operator în PF).

Se include Event Structure în DB. Implicit există cazul '[0] Timeout'. Se adaugă cu comanda 'Add Event Case' un nou caz pentru tratarea unui eveniment de tipul 'Value Change' - caz înregistrat static (Static Event Registration). Se va urmări schimbarea valorii controlului String. In fereastra de configurare setăm... La apariţia evenimentului se execută codul din cazul asociat vizualizat în figură.

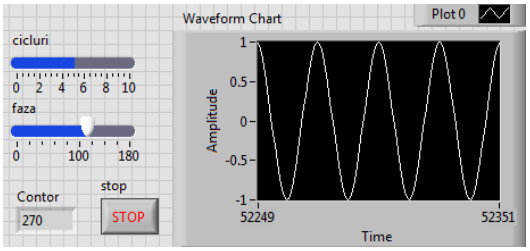




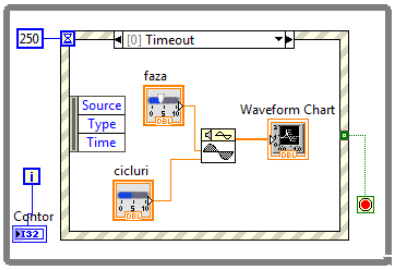
Lansăm aplicaţia. Se execută repetitiv ciclul While pe cazul '[0] Timeout' de fiecare dată după aşteptare de 20ms. Dacă se tastează un şir caractere sau modifică un şir existent (fară Enter) în controlul String urmat de Click pe suprafaţa PF are practic loc evenimentul. Acest eveniment declanşază execuţia codului din caz [1]: se afişează în indicatorul OldVal vechiul şir care a fost modificat; se execută secvenţa indice '0' din structura Sequence în care LEDul indicator String Changed se luminează (T) pentru 200ms după care în secvenţa indice '1' LEDul este stins (F) şi se iese din Event Structure.

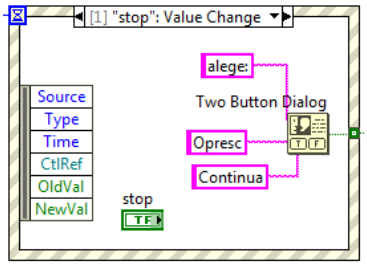
Asteptarea apariţiei unui eveniment (schimbare şir caractere) durează 20ms. Dacă nu s-a schimbat şirul se execută evenimentul 'Timeout' după care se părăseşte Structura event şi se repetă ciclul While dacă Butonul STOP nu-i pus pe T (apăsat). Se poate adăuga indicator pentru contorul ***i*** pentru a observa ciclările.

3.4. Grafic sinus + Buton 2 decizii

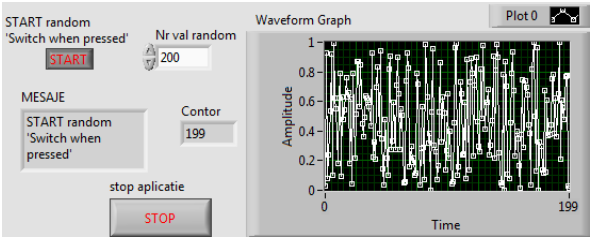


După aşteptare de 250ms se execută cazul [0] Timeout şi se actualizează repetat graficul sinus. Dacă se apasă Stop, acest eveniment declanşează cazul [1] şi se alege între Opresc şi Continuă.



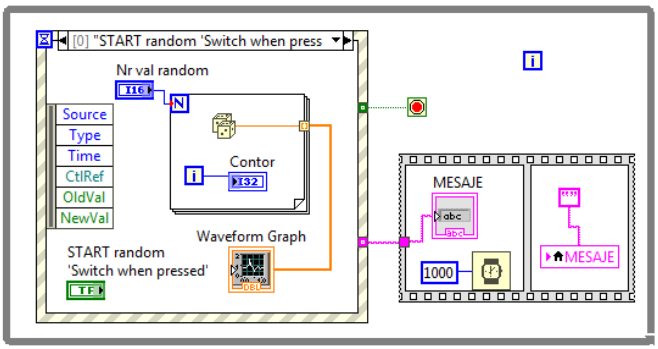


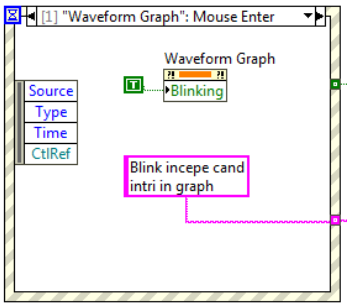
3.5. Evenimente diverse



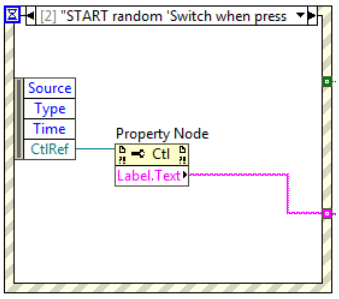
Evenimente:
[0]:La apăsare START (schimbare T→F sau F→T) se generează şi afişează 200 valori random;
[1]: cand Mouse intră în zona indicatorului grafic 'Waveform Graph', graficul începe 'Blink';

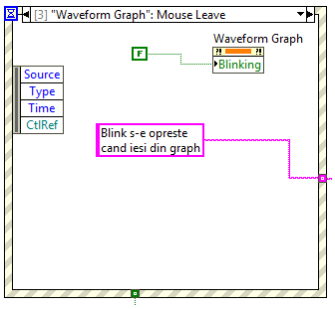
[0] "START random 'Switch when pressed'": Value Change
[1] "Waveform Graph": Mouse Enter
[2] "START random 'Switch when pressed'", "stop aplicatie", "Nr val random": Mouse Enter
[3] "Waveform Graph": Mouse Leave
[4] "stop aplicatie": Value Change

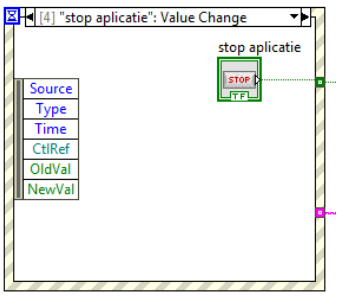




[2] cand Mouse intră în zona controlului *'Start random Switch when pressed'* sau în zona controlului '*stop aplicatie*' sau '*Nr val random*' se afişează eticheta (CtlRef→Label.Text) în indicatorul şir de caractere MESAJE pentru o secundă. Evenimentul returnează CtlRef iar PropertyNode citeşte o proprietate a unei referinţe care poate fi obiect, clasă, metodă, VI (aici Property for Control Class);

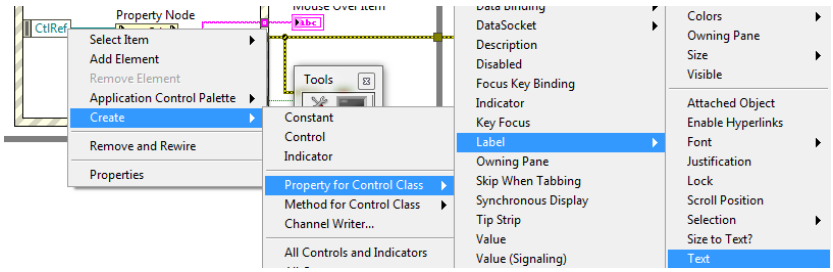






[3] când Mouse iese din zona indicatorului Waveform Graph se opreşte grafic blink;

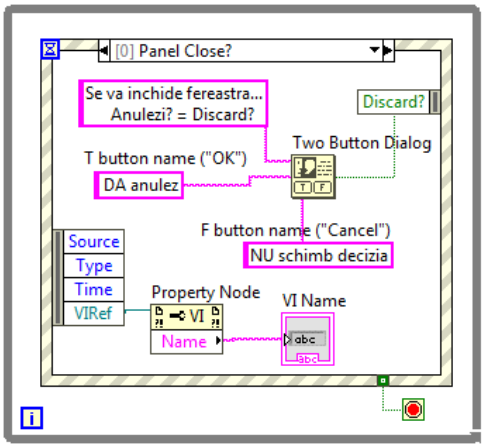
[4]: la apăsare buton STOP se opreşte ciclul While şi aplicaţia curentă. Secvenţa indice 0 (din structura Sequence) afişează mesajul pentru 1 secundă după care în secvenţa următoare (indice 1) mesajul este şters prin afişare şir vid.



3.6. **Eveniment de filtrare (Filter Event)... Panel Close?**

A apărut un eveniment şi anume Click pentru închidere panou PF. Acţiunea de închidere efectivă a ferestrei poate fi anulată prin tratarea evenimentului [0] Panel Close?. Acţiunile pot fi anulate sau modificate.

[0] Panel Close? -> eveniment de filtrare



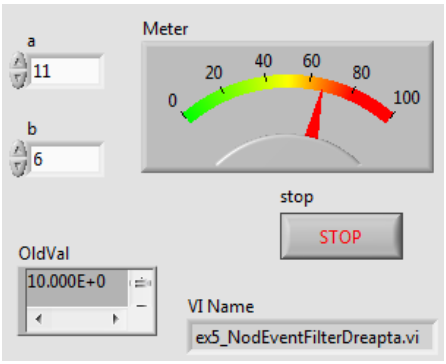
1.Cu mouse se acţionează pentru închiderea ferestrei PF a aplicaţiei.

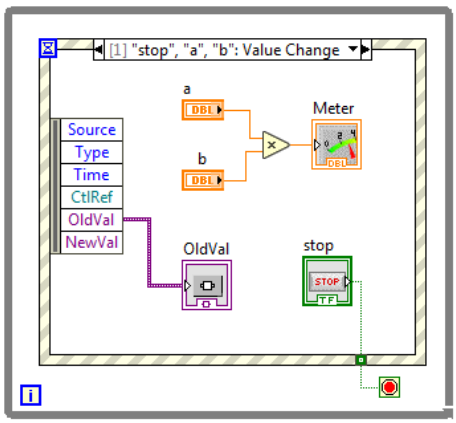
2.Prin **VIRef** din Nodul Event Data se extrage numele aplicaţiei curente. Numele se afişează în indicatorul şir caractere VI Name.

3.Prin câmpul **Discard?** al nodului Event Filter se poate anula comanda de închidere a ferestrei dacă se alege Da anulez sau se acceptă decizia de închidere aplicaţie dacă se alege NU schimb decizia.

[1] "stop","a","b": Value Change -> eveniment de notificare

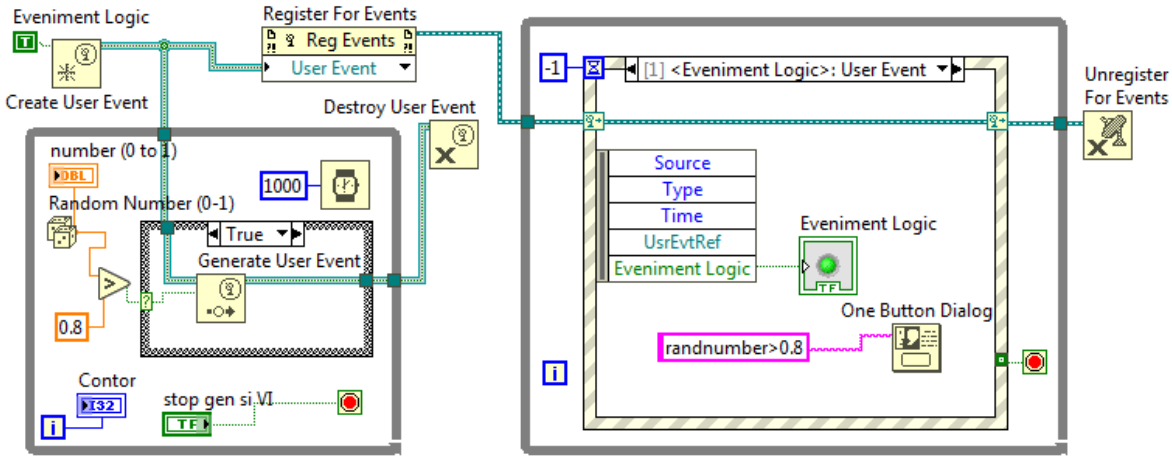
1.Se modifică valoarea controlului a sau b sau stop. Prin articolul OldVal din nodul Event Data se afişează valoarea veche. Indicatorul OldVal este de tip variant pentru că va afişa atât valori numerice (a,b) cât şi logice (False).





3.3. Eveniment dinamic declanşat prin program din Diagramă (nu din PF) fără intervenţie User în PF

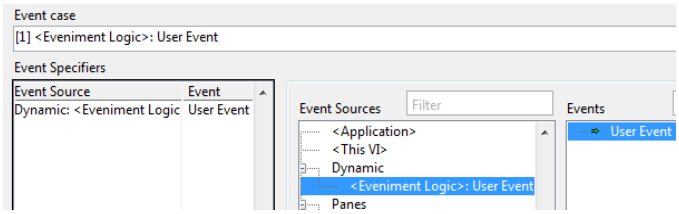
In aplicaţie sunt generate repetitiv numere aleatoare între 0 şi 1. Dacă numărul aleator curent este mai mare decât 0.8 se declanşează un eveniment pentru care se afişează mesajul ***'randnumber>0.8'*** în PF. La generarea evenimentului şi tratarea lui coparticipă funcţiile (paleta Dialog&User Interface/ Events): Create User Event care cu ieşirea ***user event out*** se leagă la funcţiile Generate User Event şi Register For Events. Evenimentul aşteptat trebuie să aibă **tip** şi **nume**. Prin intrarea **user event data type** a funcţiei Create User Event se stabileşte Numele evenimentului (***Eveniment Logic*** dat de utilizator) şi Tipul de dată al evenimentului (***Logic*** în acest caz) (valoarea True primită, nu are importanţă).



Funcţia **Create User Event** (paleta Events) returnează o referinţă la un User Event.

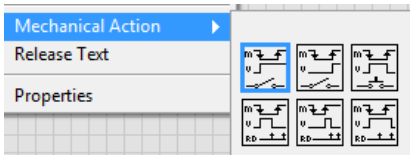
Funcţia **Register For Events** înregistrează evenimentul în cadrul structuri Event (care are în exemplu 3 cazuri).

Funcţia **Generate User Event** trimite evenimentul la toate structurile de tip Event Structure care au fost registrate pentru acel eveniment sau care conţin **un caz** pentru tratarea acestui eveniment.

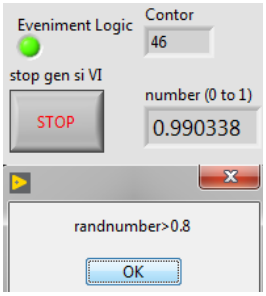


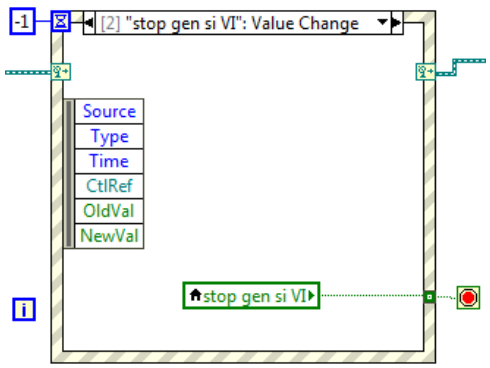
Se creează în diagramă o structură Event cu 3 cazuri:

1. Cazul '[0] Timeout' există implicit; legarea la -1 =aşteptare fără sfârşit. Se activează/bifează Show Dynamic Event Terminal la care se leagă ieşirea 1 a funcţiei Register For Events astfel încât la crearea unui nou caz să apară în fereastra de editare posibilitatea de a creea un Eveniment dinamic.



2. Cu comanda Add Event Case... se creează al doilea caz **'[1]<Eveniment Logic>:User Event'**. In fereastra deschisă (Event case, vezi Fig.) se selectează Event Sources: Dynamic/... Apoi se tratează cazul dynamic: se creează indicatorul Eveniment Logic şi se creează mesajul **randnumber>0.8** care alimentează funcţia One Button Dialog.





3. Cu comanda Add Event Case... se creează al treilea caz (caz înregistrat static) numit **'[2] 'stop gen si VI': Value Chance'.** Cazul urmăreşte evenimentul Value Chance a butonului STOP (cu eticheta 'stop gen si VI' din PF) schimbare care apare la apăsarea butonului. Apăsarea butonului (generează val. True) are acţiune dublă:

a) opreşte ciclul While de generare numere aleatoare + generare eveniment dynamic şi b) se execută cazul [3] din Event Case pentru oprirea ciclului While care conţine Event Structure cu 3 cazuri; oprindu-se astfel întreaga aplicaţie sau VI.

Event Structure este pusă în ciclul WHILE deoarece după apariţia unui eveniment structura Event Structure se consumă (termină) şi trebuie din nou executată pentru a trata un următor eveniment.

Producerea evenimentului dynamic: când se generează un număr aleator mai mare decât 0.80 se selectează cazul True (Instr. Case) în care are loc apelul funcţiei **Generate User Event**. Generarea de numere aleatoare are loc repetitiv cu temporizare de 1000 ms. Se rămâne în ciclul While până la apăsare Buton 'stop gen si VI' situaţie în care se opreşte şi aplicaţia.

Generarea periodică de numere aleatoare se întrerupe numai cu buton STOP din Panoul frontal a cărui Mechanical Action este pusă pe poziţia (1,1) în matricea de acţiuni posibile (Fig.). Numai pe acest mod se poate creea variabilă locală pe butonul de STOP şi pune variabila locală în cazul al 3-lea pentru oprire ciclul While pe Stop if true când se apasă STOP. Se apasă STOP care generează True după care se apasă din nou STOP pentru revenire la buton neapăsat pentru a genera False şi a nu se închide aplicaţia la un nou Run.