Budowa i oprogramowanie komputerowych systemów sterowania

Wykład 5

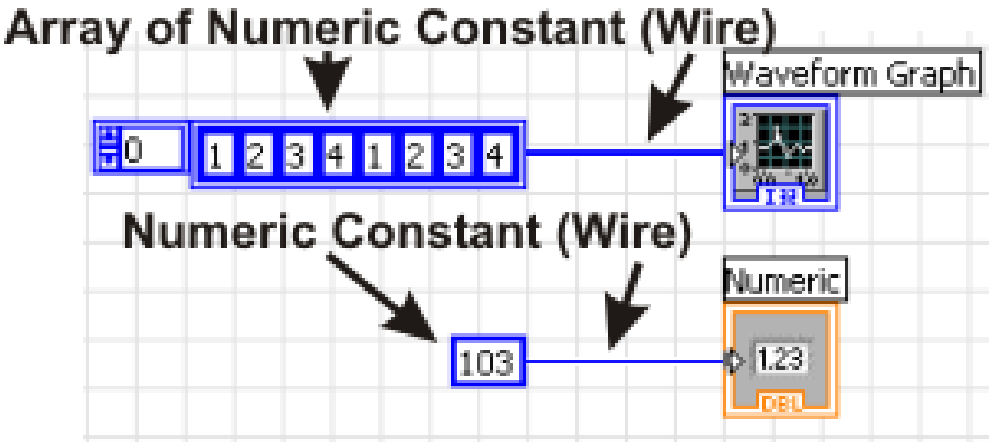
Labview 2

LabVIEW - Arrays

- Przy programowaniu w językach takich jak C czy Pascal – jednym z ważnych typów danych są struktury i tablice.
- Analogicznie tworzenie bardziej skomplikowanych VI bez wykorzystania podobnych mechanizmów byłoby niezwykle trudne.
- W Labview dostępne są tablice (Arrays) i klastry (Clusters).
- Array można porównać do tablicy w C czy też w Pascalu.
- Cluster jest strukturą w języku C, a rekordem w Pascalu, zaś strukturą .
- Array - tablica grupuje elementy jednego typu (np. tekst, wartość stałą, itp).; tablica może być wielowymiarowa
- Cluster - klaster grupujący elementy różnych typów (tekst, wartości stałe - w jednym klastrze jednocześnie)

LabVIEW - Arrays

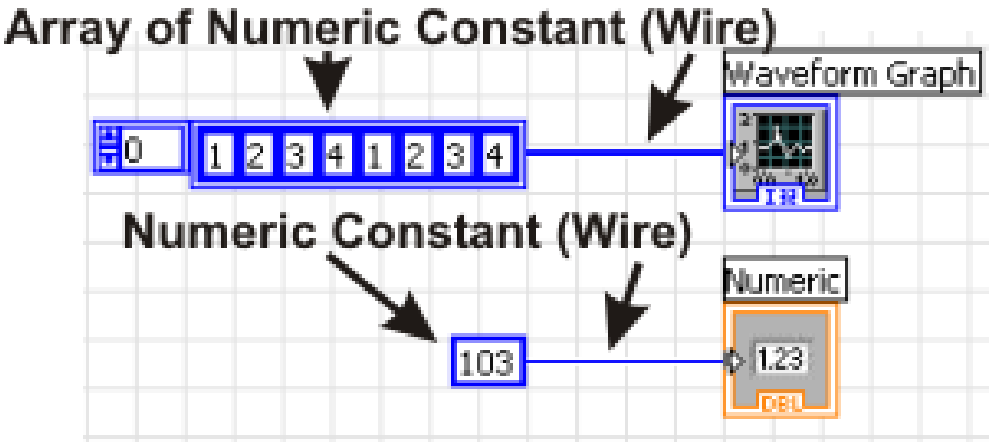
- Przykład



- W celu stworzenia diagramu - wybieramy stałą tablicę: All Functions > Array > Array Constant.
- Po umiejscowieniu ikony w diagramie blokowym - będzie ona pusta.
- Oznaczać to będzie, że nie ma danych oraz że nie ma przypisanego typu danych.
- Aby dokończyć - należy wybrać stałą z palety funkcji: Arith/Compare > Numeric > Num Const.

LabVIEW - Arrays

- Przykład



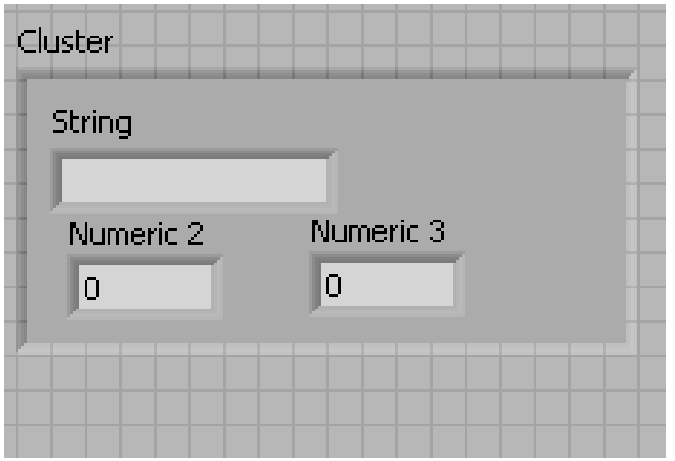
- Stałą przeciągamy do środka ikony tablicy. Następnie powiększamy (rozciągając bok) rozmiar tablicy do żądanego.
- Przewód (Wire) zawierające tablicę zgrupowanych danych jest koloru takiego samego jak dane które grupuje. Różnica leży w grubości połączenia - Array grupujący typ danych A - jest grubszy. Grubość połączenia wzrasta ze wzrostem ilości wymiarów tablicy.
- Przewód (Wire) symbolizujący połączone klastry - jest koloru różowego lub brązowego (w zależności od składników klastru).

LabVIEW - Clusters

- Klaster jest elementem podobnym do struktury w języku C, czy rekordu w Pascalu.
- Pozwala na zgrupowanie kilku różnych typów danych.
- Przykładowy klaster może zawierać liczby zmiennoprzecinkowe (double), łańcuchy znakowe (string), liczby całkowite (integer), oraz inne dane jednocześnie.
- Istotna jest kolejność występowania elementów w klastrze.
- Jeżeli kolejność występowania elementów nie jest zachowana pomiędzy funkcjami - wówczas nie możemy połączyć funkcji.
- Podstawowe funkcje realizowane przez klaster przedstawione są w poniższym przykładzie.

LabVIEW - Clusters

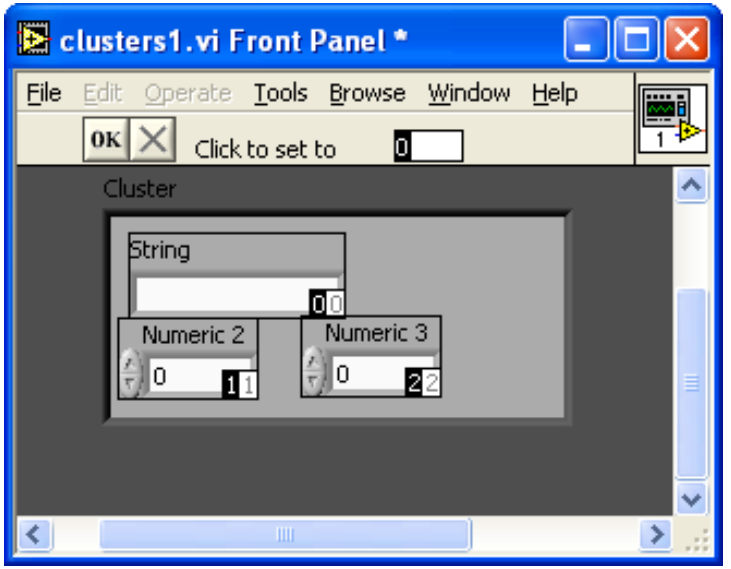
- Przykład



- Na panelu frontowym wybieramy All Controls > Array & Cluster > Cluster i umieszczamy na panelu.
- W pole nowo dodanego klastra dodajemy wskaźnik tekstu: (String Indicator) Text Inds > String Ind. Umieszczamy również wewnątrz klastra - wskaźniki wartości (również w pole klastra). Num Inds > Num Ind

LabVIEW - Clusters

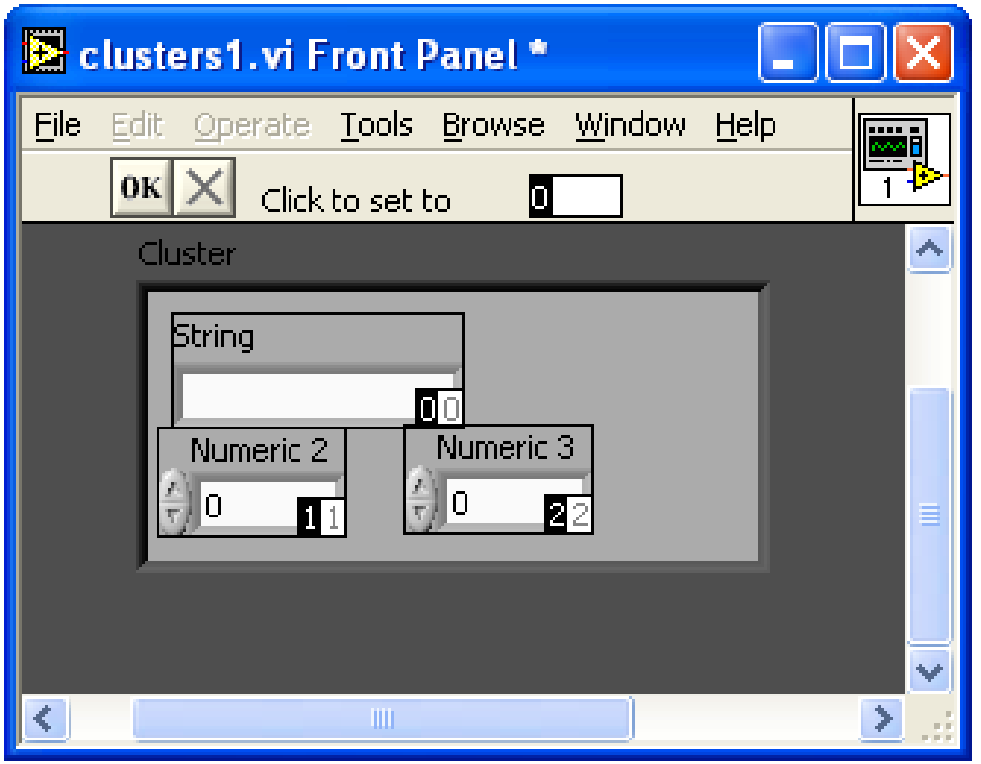
- Przykład



- W celu ustalenia kolejności w klastrze:
  - kliknij prawym przyciskiem na krawędź klastra
  - wybierz Reorder Controls in Cluster
  - uzupełnij pole Click To Set wartością 0.
  - kliknij na pole które ma być pierwsze (Numeric 2)
  - kliknij na pole które ma być drugie (Numeric 3)
  - kliknij na pole które ma być trzecie (String)
  - zaakceptuj - klikając napis OK

LabVIEW - Clusters

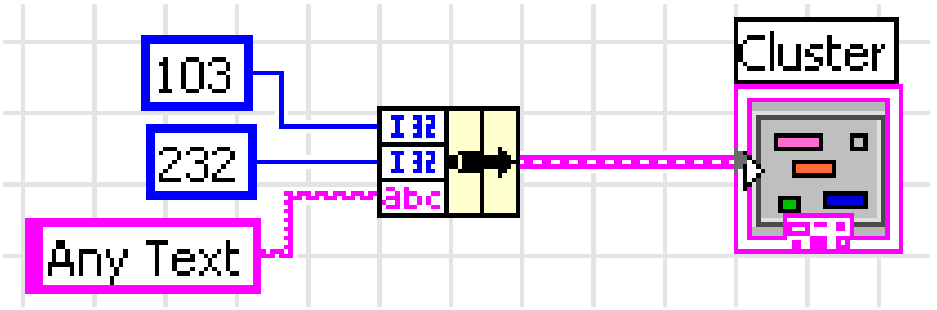
- Przykład



- W celu ustalenia klastra na panelu frontowym - jako wskaźnika, należy kliknąć prawym przyciskiem na krawędź klastra i wybrać Change to Indicator.

LabVIEW - Clusters

- Przykład



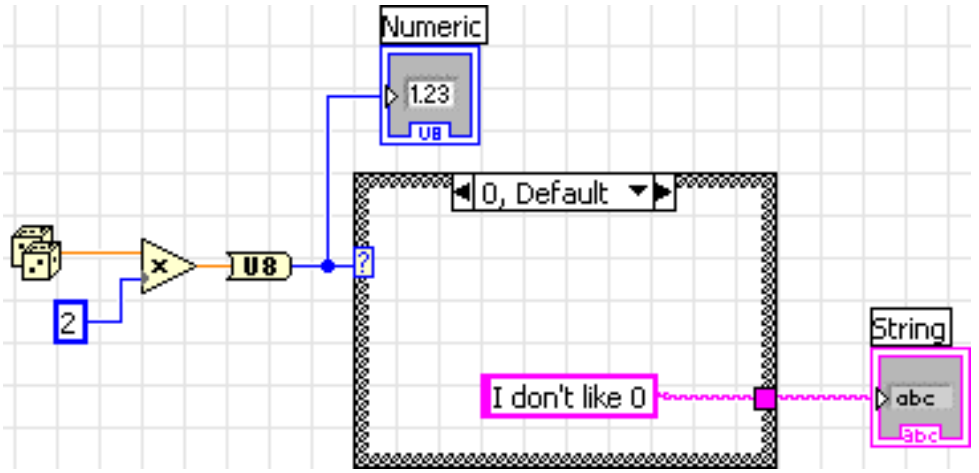
- Na diagramie blokowym z palety funkcji wybieramy All Functions > Clusters > Bundle. Element ten grupuje wchodzące z lewej strony dane i tworzy z niego klaster. Klaster ten zostanie przesłany do wskaźnika - zgodnie z przedstawionym diagramem.
- Należy zwrócić szczególną uwagę na kolejność podłączania elementów po lewej stronie. Pierwsze dwa elementy to Numeric Constant. Trzeci z nich to String Constant (All Functions > String > String Constant)

LabVIEW - Cases

- Kolejne opisywane poniżej elementy języka - pozwalają na kontrolowanie przebiegu programu.
- Struktura case pozwala wybrać działanie w zależności od wartości.
- Na przykładzie zilustrowane zostanie działanie pętli case.
- Warunek case będzie określony dla wartości 0, 1, 2.
- W każdym z tych trzech przypadków zostanie wypisany inny komunikat.

LabVIEW - Cases

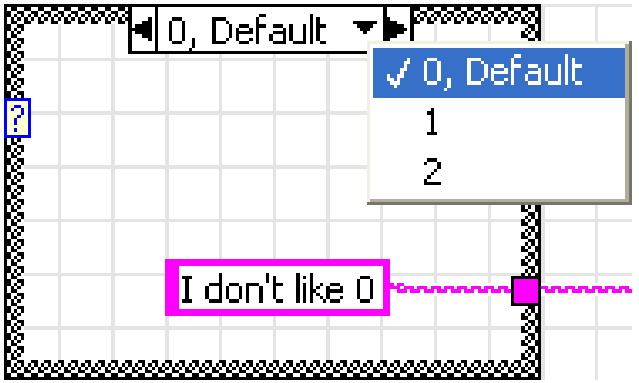
- Przykład



- W celu wybrania struktury case należy z palety funkcji wybrać Exec Ctrl > Case Structure.
- Case działa w oparciu o zasadę wyboru algorytmu. Wybór ten zależny jest od wartości sterującej strukturą case. W rozwinięciu warunku case - otrzymamy na wyjściu struktury poniższe łańcuchy.

LabVIEW - Cases

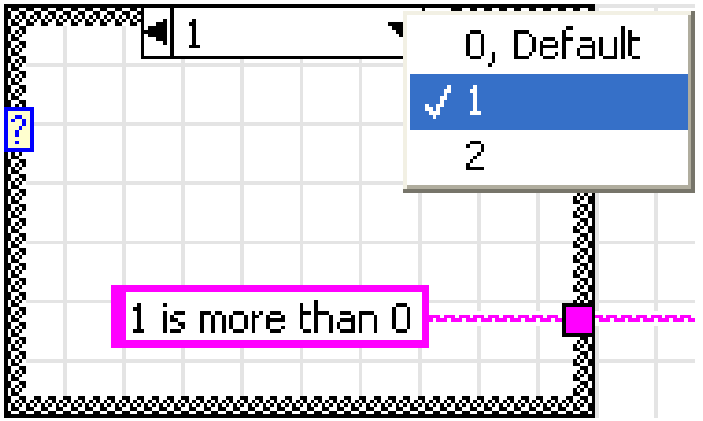
- Przykład



- W przypadku wartości 0, oraz w przypadku gdy uzyskamy wartość różną od (0,1,2) - zwrócony zostanie łańcuch (string) "I don't like 0"

LabVIEW - Cases

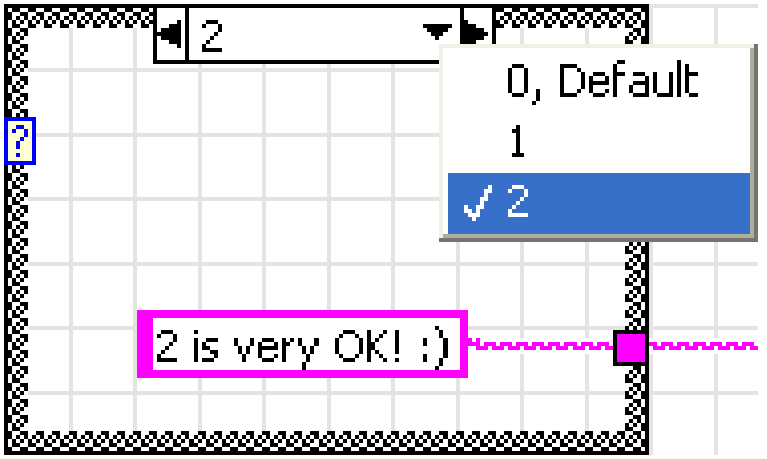
- Przykład



- W przypadku wartości 1, zwrócony zostanie łańcuch "1 is more than 0"

LabVIEW - Cases

- Przykład



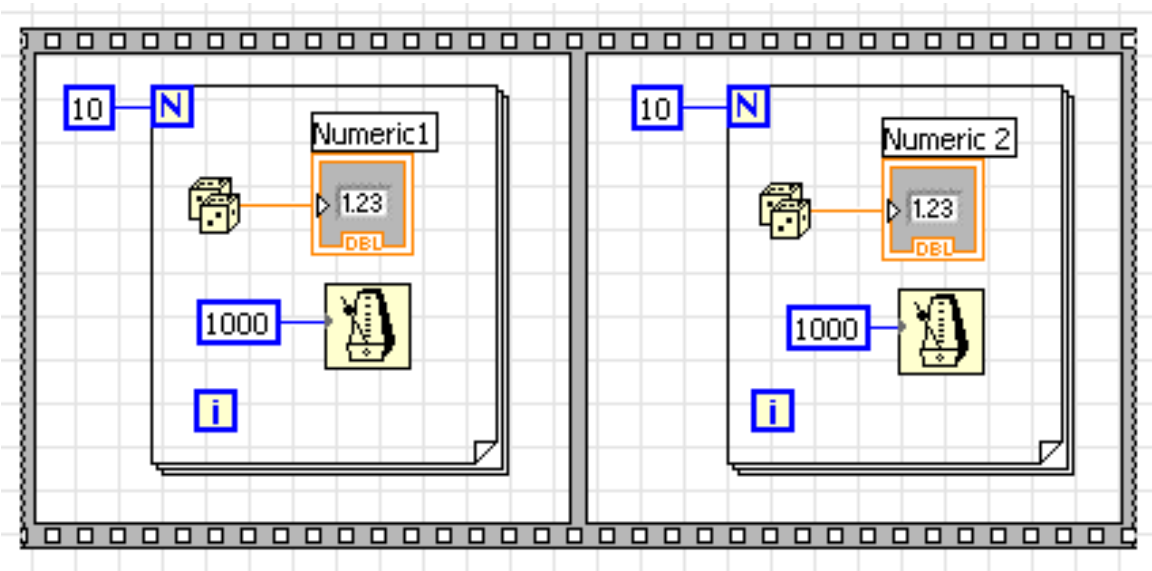
- W przypadku wartości 1, zwrócony zostanie łańcuch "2 is very OK!"

LabVIEW - Sequence structures

- LabVIEW zakłada że bloki które nie są połączone ze sobą - mogą być wykonywane jednocześnie.
- Niekiedy jednak zdarza się - że procesy powinny zostać wykonywane w ściśle określonym porządku.
- Przykładem może być chociażby transmisja z zewnętrznym urządzeniem. Ściśle określony protokół będzie narzucać wymóg zachowania określonej sekwencji wymienianych danych.

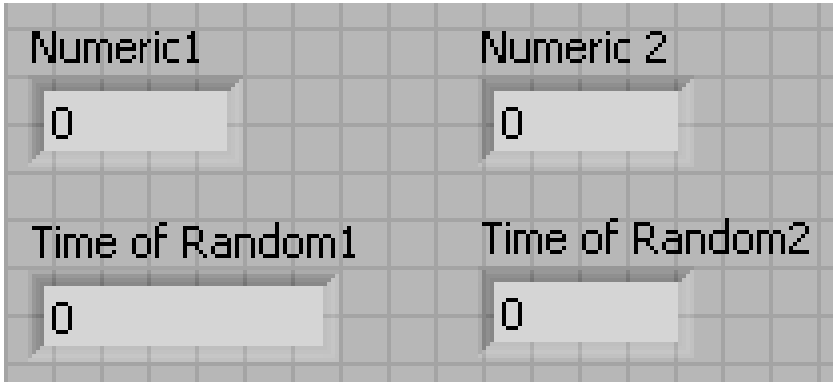
LabVIEW - Sequence structures

- Rozwiązaniem jest struktura sekwencyjna (Sequence structure). Jest ona dostępna w palecie funkcji Exec Ctrl > Flat Sequence.



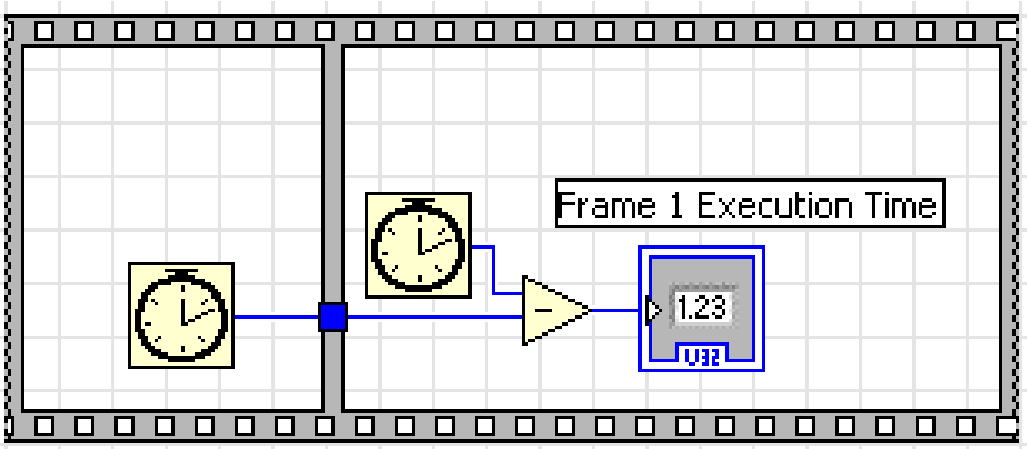
LabVIEW - Sequence structures

- Przykład
- Będziemy mierzyć czas wykonania kolejnych elementów sekwencji. Zmierzymy czy czas losowania 10 pierwszych liczb rzeczywiście wynosi 10 * 1000ms. Wynik zostanie przesłany do wskaźnika numerycznego (Numeric Indicator) o nazwie "Time of Random1". Osobno zmierzymy czas losowania kolejnych 10 liczb. Ten wynik zostanie przesłany do "Time of Random2"
- Panel frontowy przybierze następującą postać.



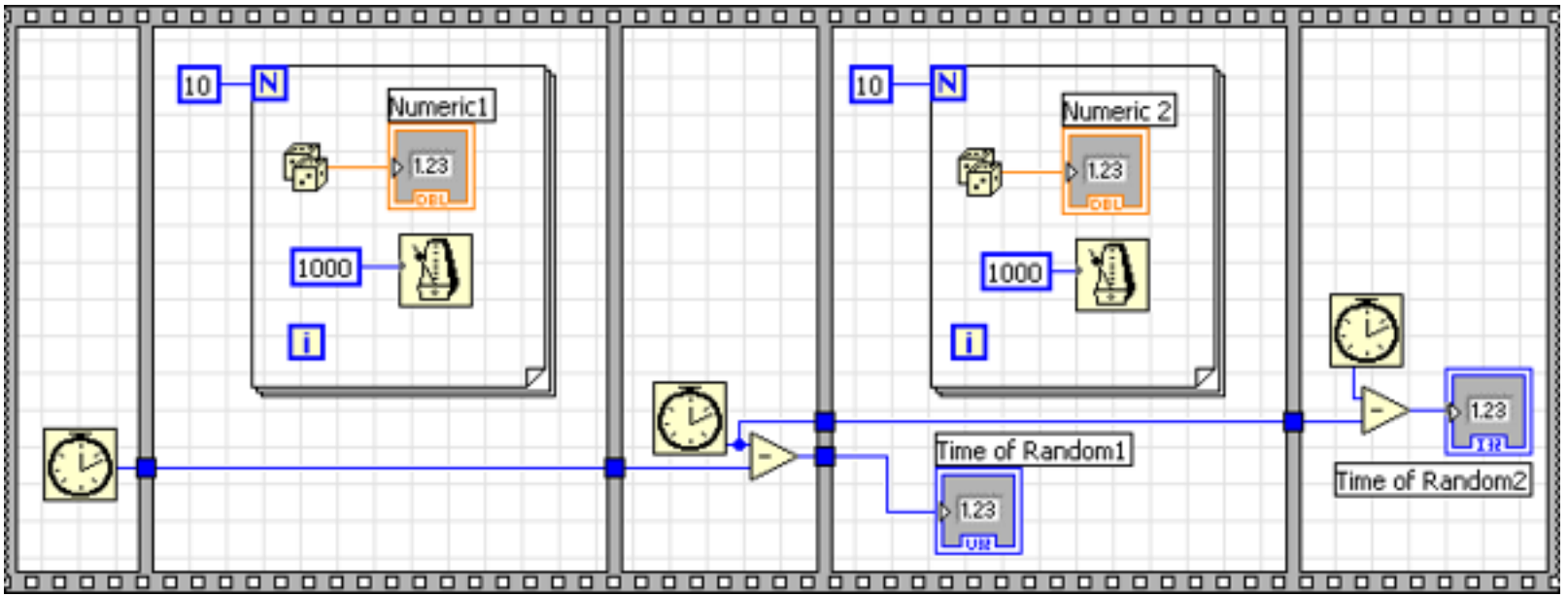
LabVIEW - Sequence structures

- Przykład
- Wskazówka: Mierzenie czasu odbędzie się w prosty sposób. Skorzystamy z następującego rozwiązania dla struktury sekwencyjnej.



LabVIEW - Sequence structures

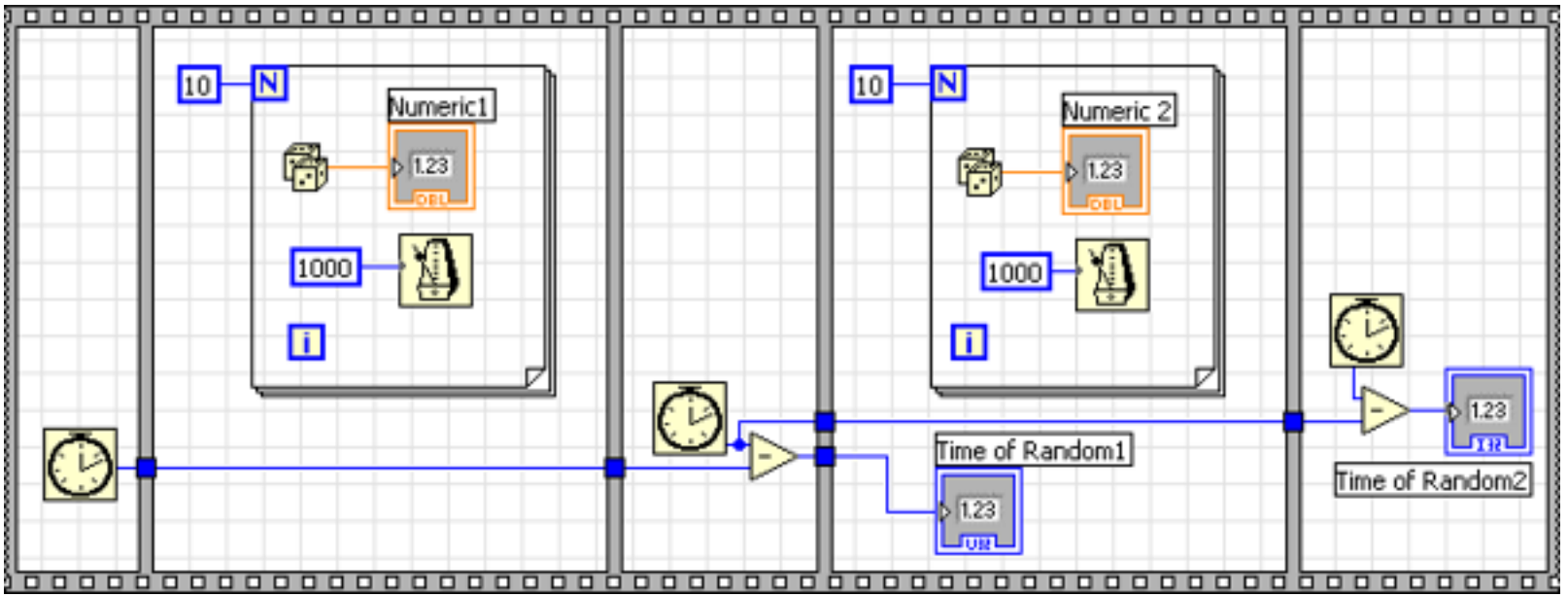
- Przykład



- Po rozpoczęciu wykonywania pierwszej ramki - zostanie pobrany czas (nazwijmy go A).
- Po zakończeniu wykonywanie pierwszej struktury (po wykonaniu wszystkich jego elementów) - rozpocznie się wykonywanie drugiej ramki. Zostanie pobrany czas (B).

LabVIEW - Sequence structures

- Przykład



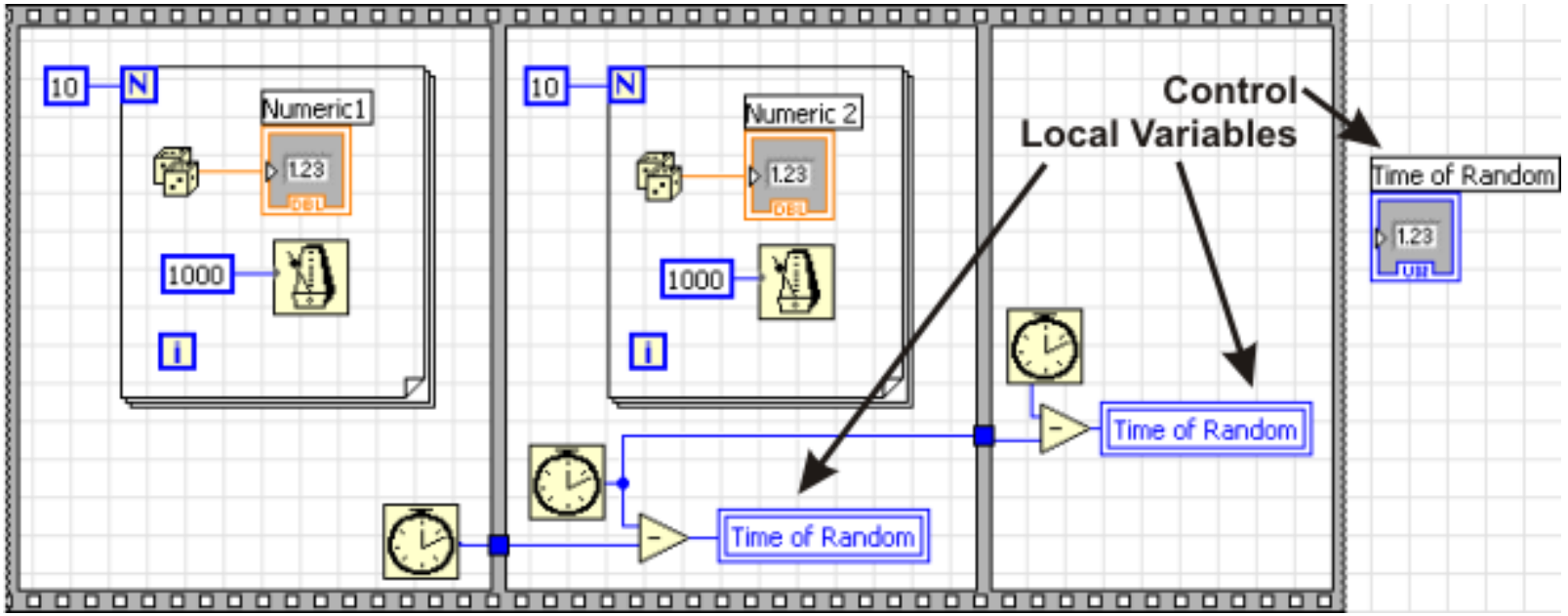
- Różnica czasów (B - A) jest niczym innym jak czasem wykonywania się ramki 1.
- Element służacy do odmierzania czasu znajduje się w palecie funkcjii: All Functions > TIme & Dialog > Tick Count (ms)

Zmienne lokalne

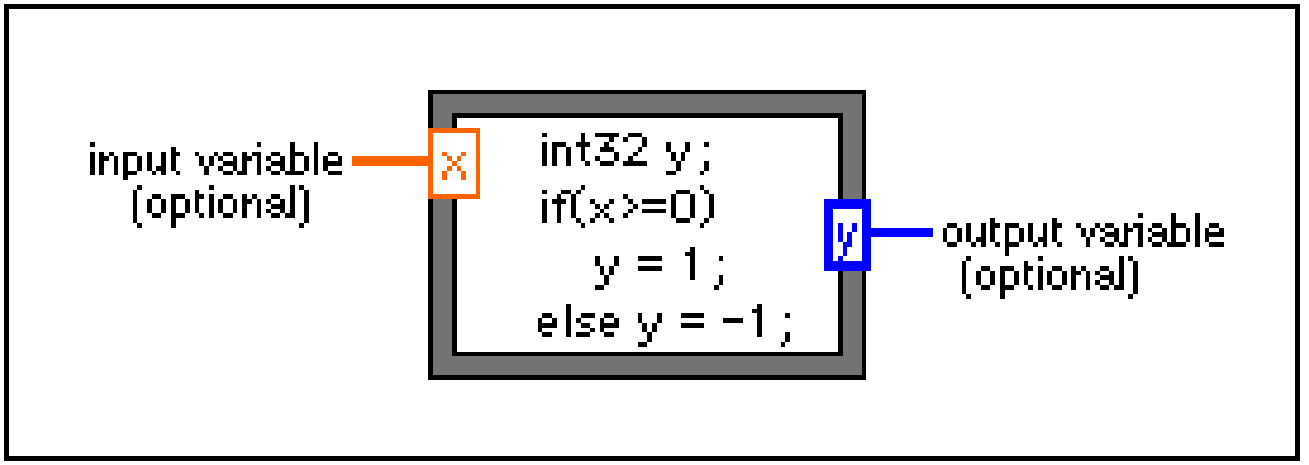
- Co jeżeli jednak będziemy chcieli do tego samego wskaźnika wpisać najpierw czas wykonywania pierwszego elementu, zaś później drugiego? Chcemy to uczynić do elementu o nazwie "Time of Random".
- Nasz wskaźnik wartości (Numeric Indicator) może być tylko w jednej ramce. Chcemy do niego odwoływać się w każdej sekwencji. Jak rozwiązać problem?
- Skorzystam ze zmiennych lokalnych (Local variables). Zamiast tworzyć "Time of Random1" i "Time of Random2" możemy utworzyć "Time of Random".
- Na podstawie "Time of Random" można utworzyć dwie zmienne lokalne (Local variables), umieszczone w miejsce poprzednich elementów. Ilustruje to kolejny diagram blokowy.

Zmienne lokalne

- Tworzymy wskaźnik (Numeric Indicator). Na diagramie blokowym klikamy prawym przyciskiem myszy. Z menu wybieramy Create > Local Variable. Czynność powtarzamy, tworząc też drugą zmienną lokalną.

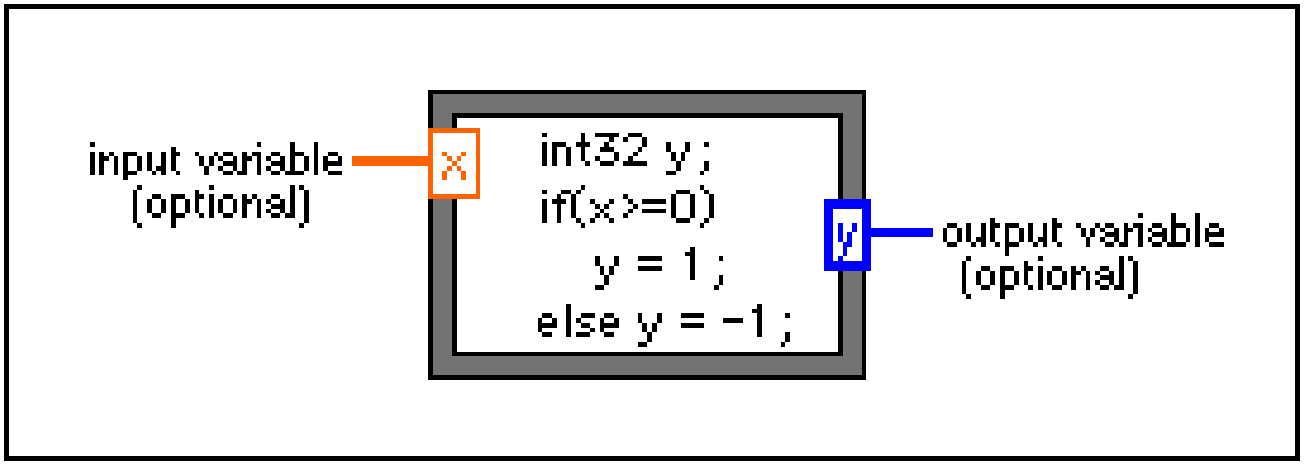


Formula Node



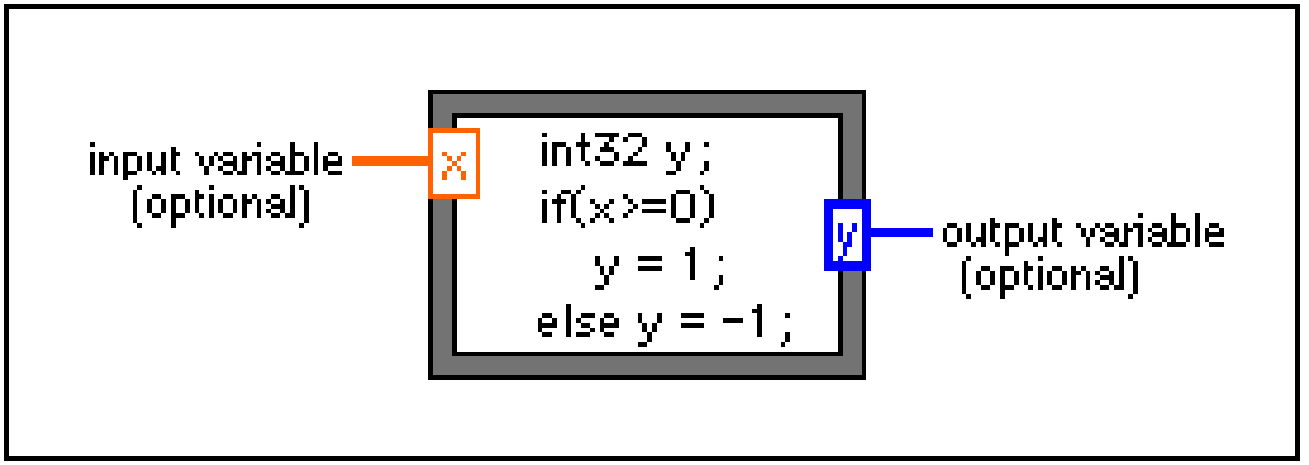
- Oblicza matematyczne formuły i wyrażenia w sposób podobny do C w ramach schematu blokowego.
- Następujące funkcje są dozwolone w wyrażeniach abs, acos, acosh, asin, asinh, atan, atan2, atanh, ceil, cos, cosh, cot, csc, exp, expm1, floor, getexp, getman, int, intrz, ln, lnp1, log, log2, max, min, mod, pow, rand, rem, sec, sign, sin, sinc, sinh, sizeOfDim, sqrt, tan, tanh.

Formula Node



- W celu utworzenia terminali wejściowych należy kliknąć prawym klawiszem na obramowaniu i wybrać Add Input z menu podręcznego.
- Wpisać nazwę zmiennej, możliwa jest późniejsza modyfikacja nazw zmiennych z wykorzystaniem narzędzi Labeling lub Operating.
- Podobnie dla terminali wyjściowych należy kliknąć prawym klawiszem na obramowaniu i wybrać Add Output z menu podręcznego.

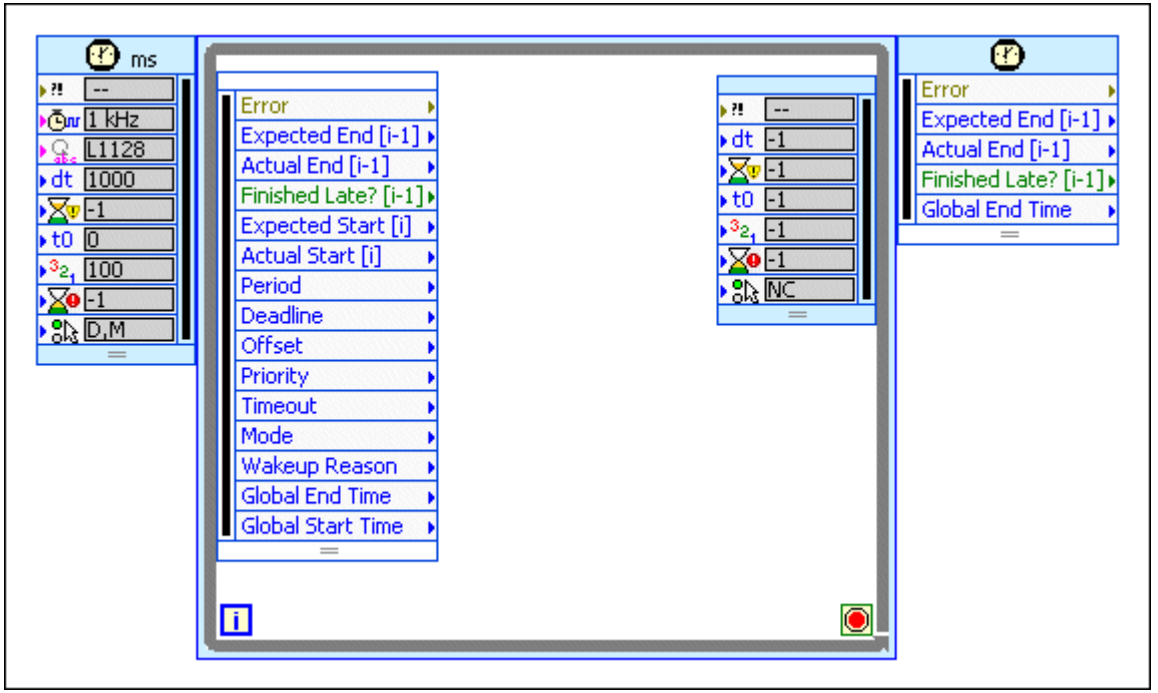
Formula Node



- Terminale są czułe na wielkość znaków. Nie ma ograniczeń na liczbę wejśc, wyjść oraz równań. Obramowanie wejść jest cieńsze od wyjść.
- Domyślnym typem danych dla wyjść jest typ zmiennoprzecinkowy podwójnej precyzji.
- Zmianę typu możemy dokonać deklarując zmienną określonego typu w bloku Formula Node, np. int32 y.

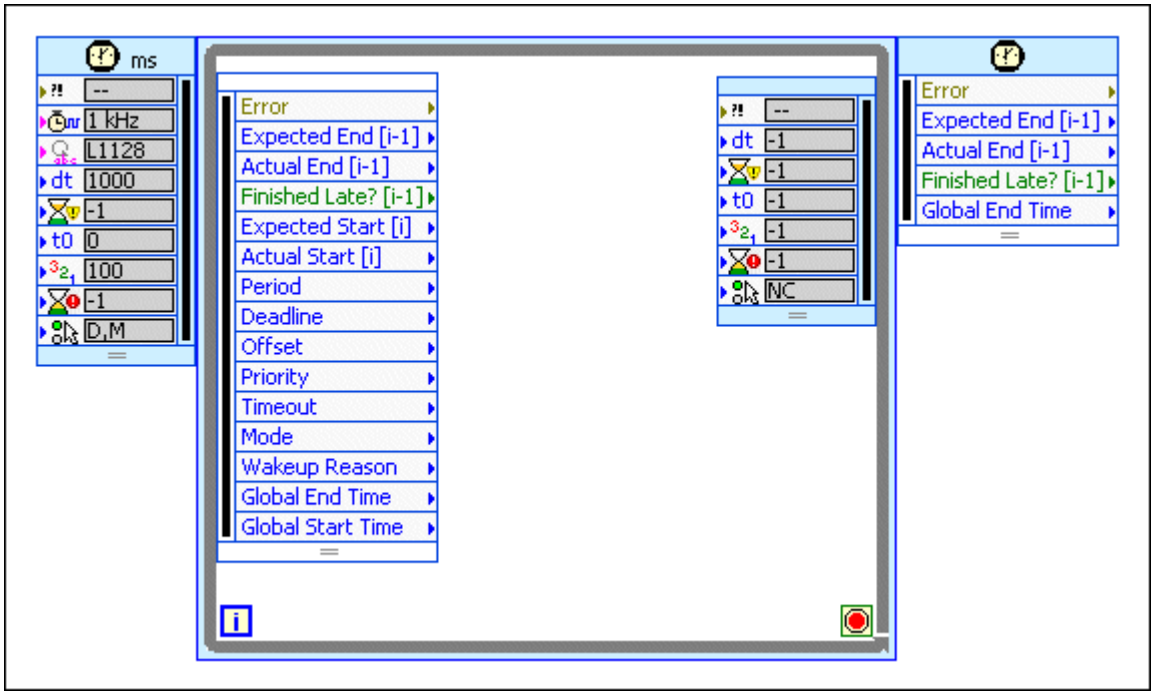
Timed Loop

- Pętla Timed Loop jest wykorzystywana w aplikacjach wymagających precyzyjnego odmierzania czasu, lub różnych poziomów priorytetów wykonywania.



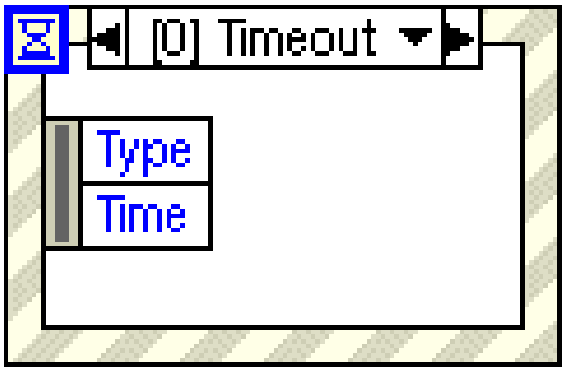
Timed Loop

- Wykonuje jeden lub więcej sub-diagramów lub ramek sekwencyjnie w każdej iteracji pętli w chwili określonej w ustawieniach pętli.
- Kliknięcie prawym klawiszem myszy na ramce pętli umożliwia dodawanie ramek.



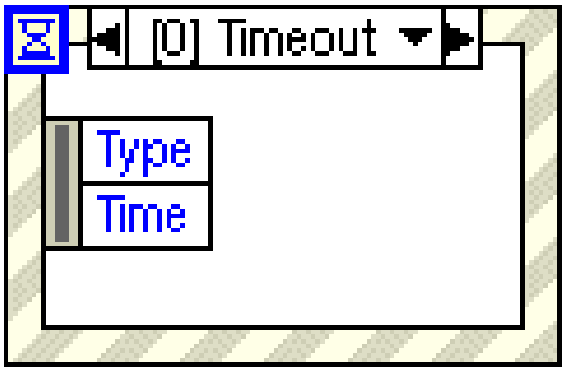
Event Structure

- Struktura Event może mieć jeden lub wiele subdiagramów lub zdarzeń, tylko jeden z nich jest wykonywany kiedy struktura jest wykonywana.
- Struktura Event czeka na pojawienie się zdarzenia i następnie wykonuje odpowiadający temu zdarzeniu diagram.



Event Structure

- Podłączenie terminala w lewym górnym rogu pozwala określić liczbę milisekund jaką struktura ma czekać na wystąpienie zdarzenia.
- Domyślne jet -1 co oznacza, że czeka w nieskończoność.
- Kliknięcie prawym klawiszem na ramce pozwala dodawać nowe zdarzenia i konfigurować które zdarzenia mają być obsługiwane.



Event Structure