Politechnika Warszawska
Wydział Transportu
Zakład Telekomunikacji w Transporcie

Podstawy użytkowania programu LabView

Opracował : mgr inż. Adam Rosiński

Wrzesień 2004

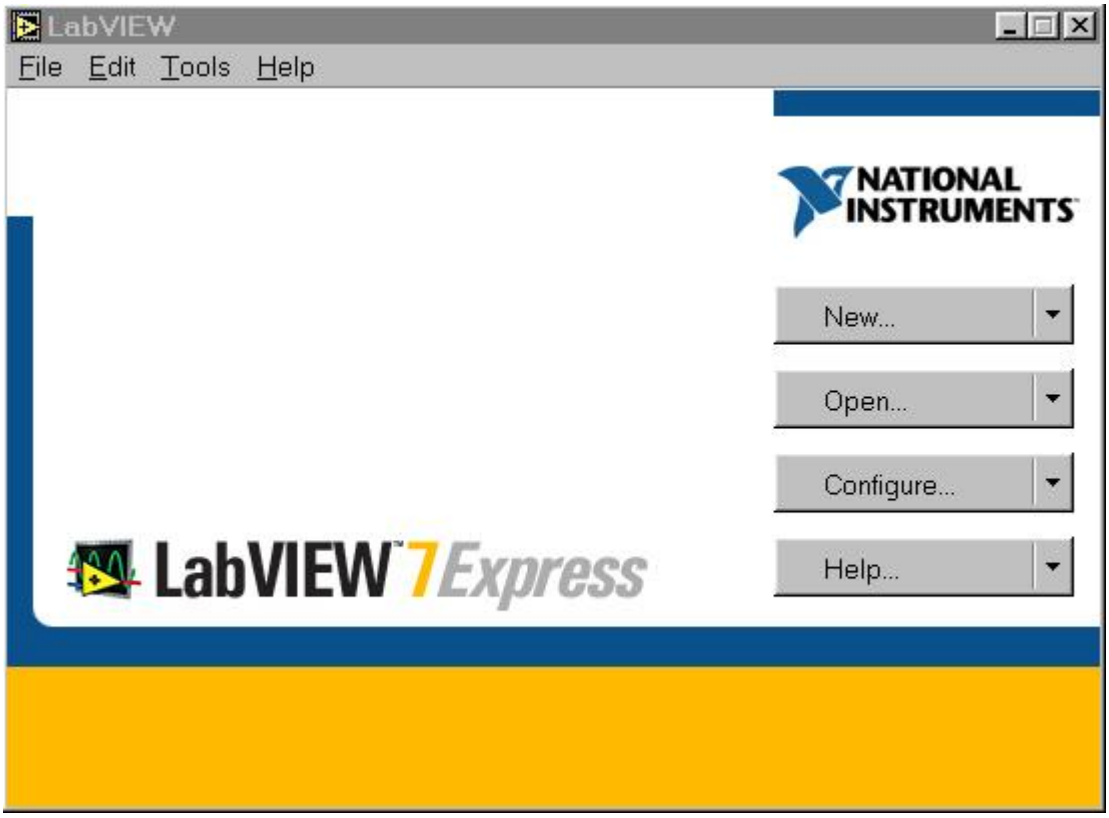
Spis treści:

1. Wstęp

Program LabView jest środowiskiem programistycznym przeznaczonym do tworzenia oprogramowania dla systemów kontrolno-pomiarowych. Wizualizacja danego procesu jest przedstawiona na ekranie monitora w postaci wirtualnego przyrządu pomiarowego, np. oscyloskopu, multimetru, zestawu wskaźników świetlnych (diod). Dlatego też nazywany jest często przyrządem wirtualnym (ang. *virtual instrumet,* w skrócie *VI*).

Po uruchomieniu programu pojawia się okno (rys. 1) w którym mamy cztery możliwości wyboru:

- utworzenia nowego wirtualnego przyrządu (*New*),
- otwarcia już istniejącego (*Open*),
- konfiguracji programu (*Configure*),
- opcje pomocy (*Help*).

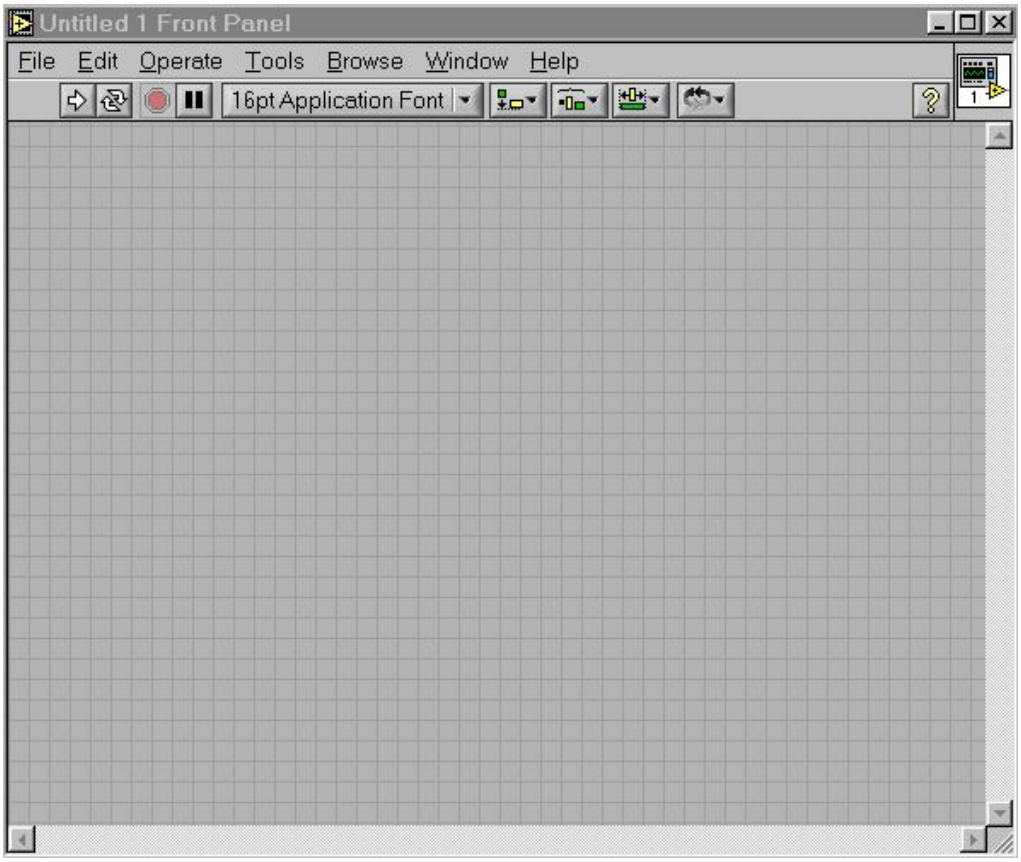


Rys. 1. Okno startowe programu LabView

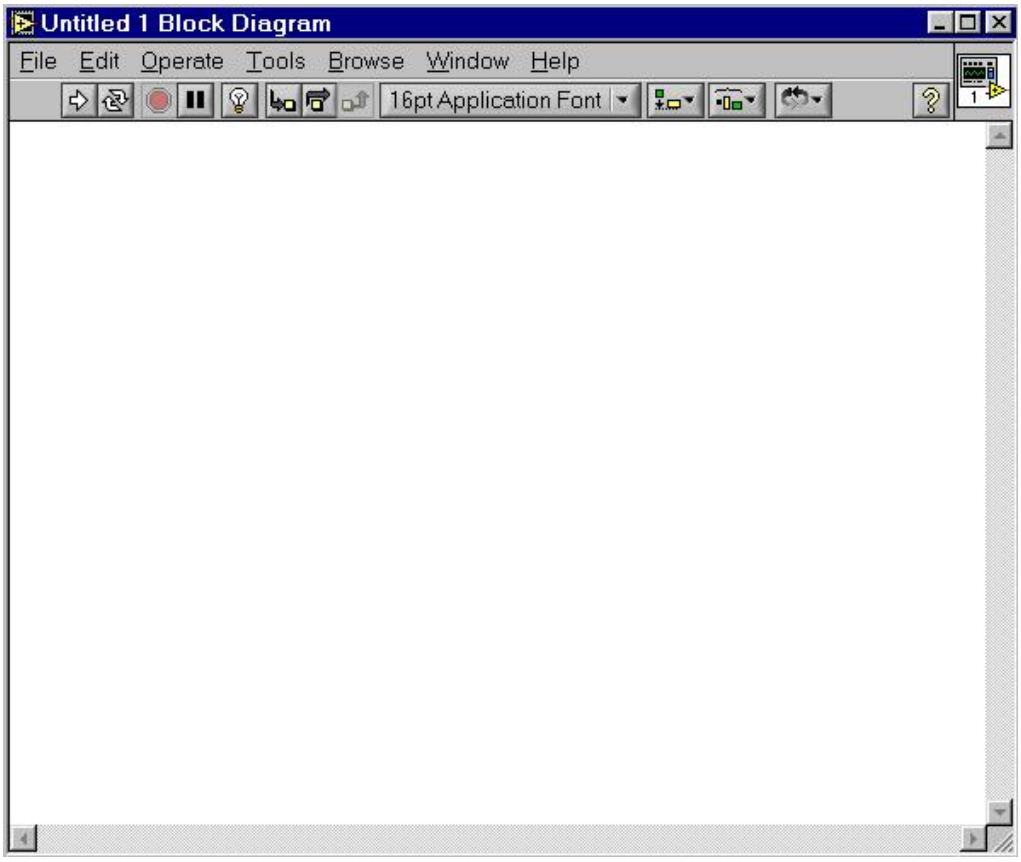
Po wybraniu opcji *New-Blank VI* pojawiają się dwa okna:

- panel frontowy (*Front Panel*) - rys. 2,
- schemat blokowy (*Block Diagram*) - rys. 3.

Stanowią one nierozłączną, powiązaną pomiędzy sobą całość.



Rys. 2. Panel frontowy

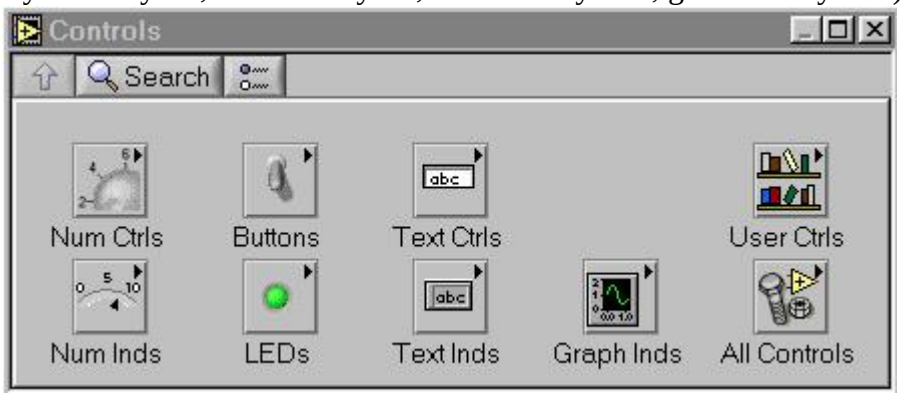


Rys. 3. Schemat blokowy

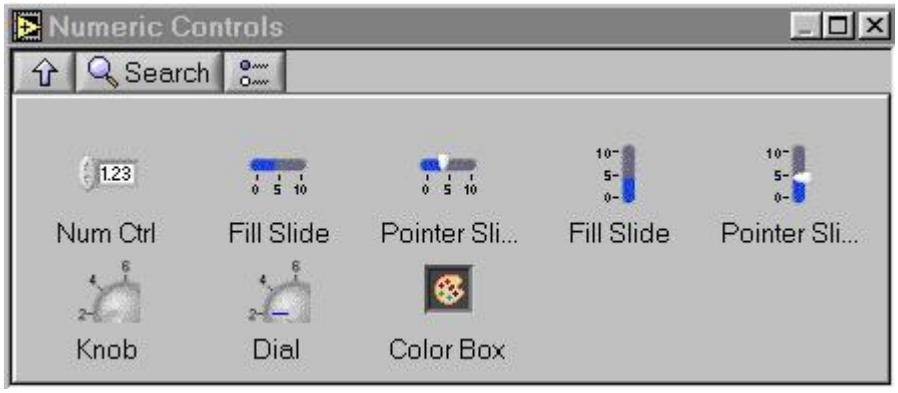
2. Panel frontowy

Okno panelu frontowego pełni rolę graficznego interfejsu między programem VI, a użytkownikiem. Przy pomocy odpowiednich elementów (np. przełączników, wyświetlaczy) możliwe jest sterowanie przyrządem, tak jakby był to przedni panel rzeczywistego przyrządu. Dodawanie poszczególnych elementów możliwe jest po wybraniu z menu górnego opcji *Window-Show Controls Palette*. Paleta kontrolek przedstawiona jest na rys. 4. Umożliwia ona wybór dwóch rodzajów elementów:

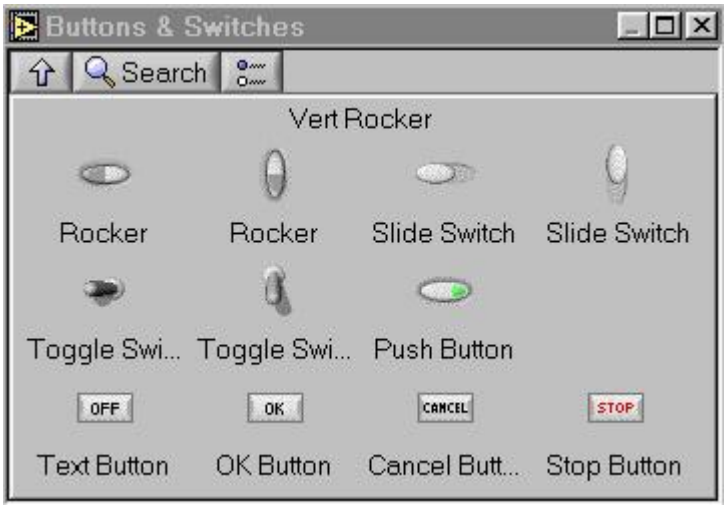
- kontrolek umożliwiających regulację wartości wejściowych programu VI (suwaki i potencjometry – rys. 5, przyciski i przełączniki – rys. 6, pola tekstowe – rys. 7),
- wskaźników przedstawiających wartości wyjściowe programu VI (wyświetlacze: numeryczne – rys. 8, diodowe- rys. 9, tekstowe- rys. 10, graficzne- rys. 11).



Rys. 4. Paleta kontrolek

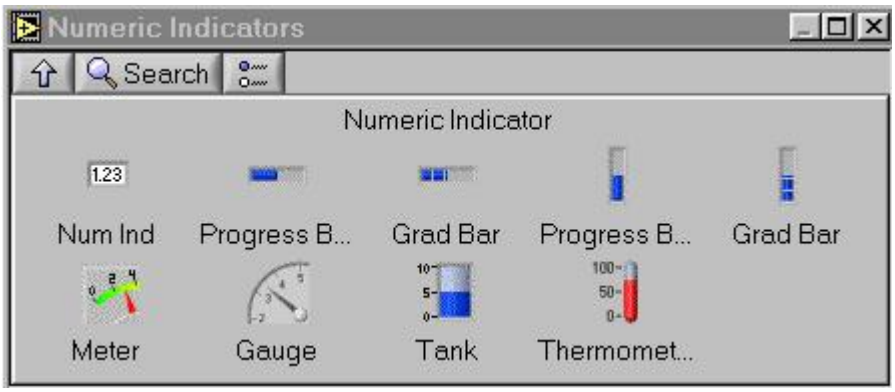


Rys. 5. Paleta kontrolek – suwaki i potencjometry



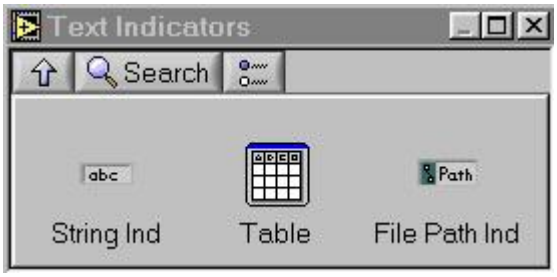
Rys. 6. Paleta kontrolek – przyciski i przełączniki

Rys. 7. Paleta kontrolek – pola tekstowe

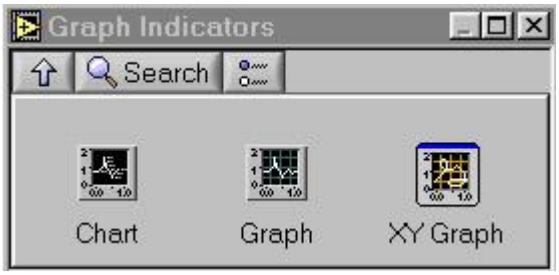


Rys. 8. Paleta kontrolek – wyświetlacze numeryczne

Rys. 9. Paleta kontrolek – wyświetlacze diodowe



Rys. 10. Paleta kontrolek – wyświetlacze tekstowe

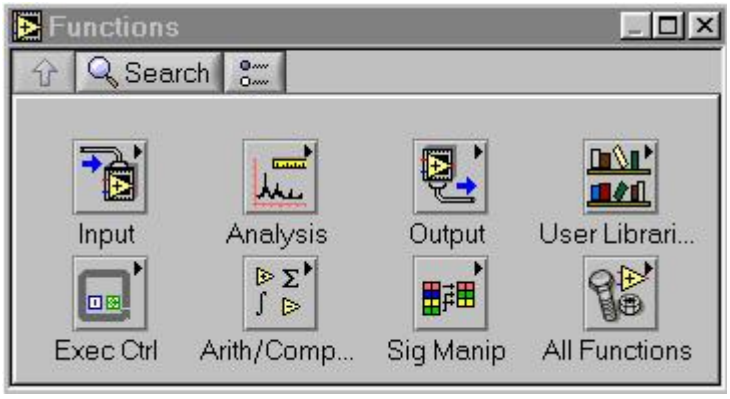


Rys. 11 Paleta kontrolek – wyświetlacze graficzne

Uwaga:
Paleta kontrolek dostępna jest tylko w oknie panelu frontowego. Dodanie elementu w tym oknie powoduje automatyczne dodanie odpowiadającego mu symbolu w oknie schematu blokowego.

3. Schemat blokowy

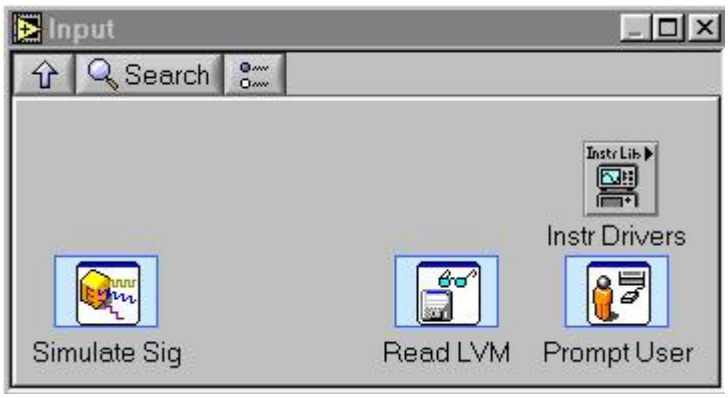
Okno schematu blokowego jest graficznym zapisem kodu programu realizującego funkcje przyrządu wirtualnego. Używa się do tego graficznego języka G, który w odróżnieniu od tekstowych języków programowania, wykorzystuje schematy blokowe. Umożliwia to szybkie stworzenie algorytmów sterujących. W oknie tym są odwzorowane wszystkie elementy jakie zostały umieszczone na panelu frontowym (wartości wejściowe i wyjściowe programu VI). Powiązania pomiędzy tymi elementami muszą odpowiadać zadaniom projektowanego przyrządu wirtualnego. W tym celu wykorzystuje się różnego rodzaju funkcje i struktury, które pozwalają stworzyć różne zależności między sygnałami wejściowymi i wyjściowymi. Dodawanie poszczególnych elementów możliwe jest po wybraniu z menu górnego opcji *Window-Show Functions Palette*.



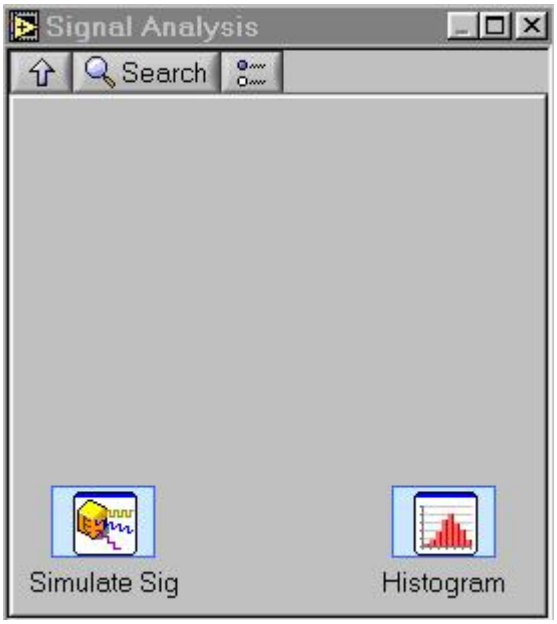
Rys. 12. Paleta funkcji

Paleta funkcji przedstawiona jest na rys. 12. Umożliwia ona wybór następujących rodzajów elementów:

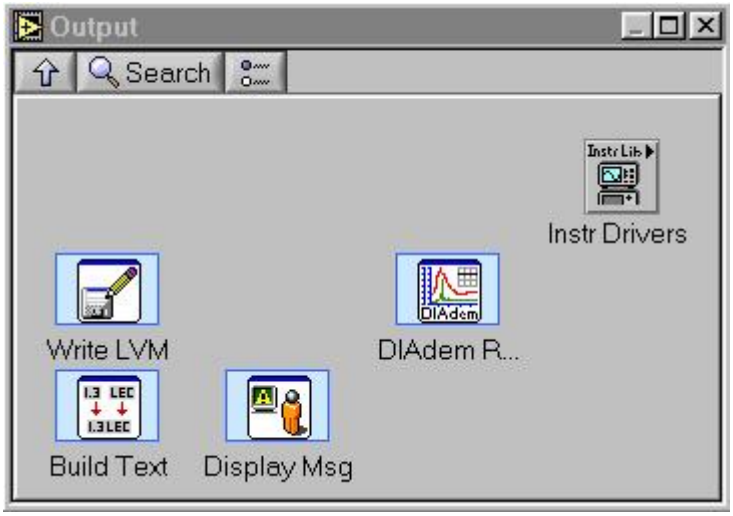
- rodzaju wejść (rys. 13),
- sposobu analizy sygnałów (rys. 14),
- rodzaju wyjść (rys. 15),
- struktur sterujących i funkcji czasowych (rys. 16)
- zależności arytmetycznych i logicznych (rys. 17),
- zmiany sygnału (rys. 18).



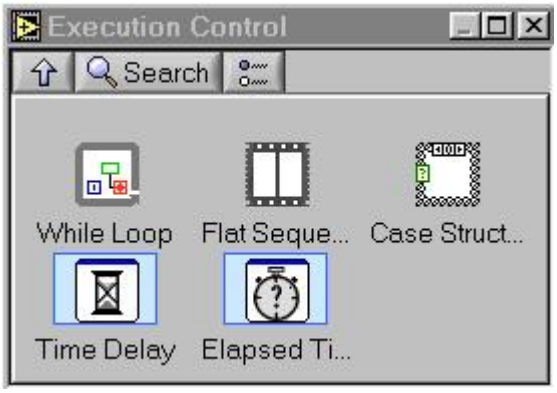
Rys. 13. Paleta funkcji - wejścia



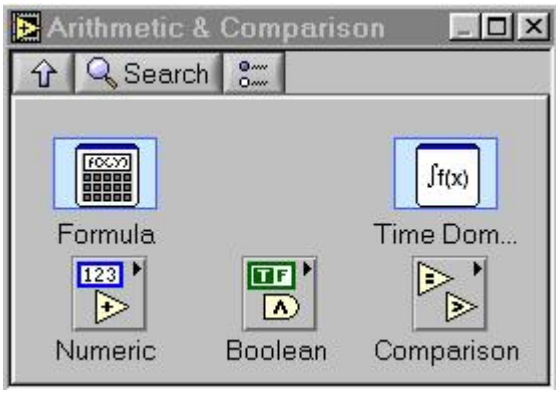
Rys. 14. Paleta funkcji – analiza sygnału



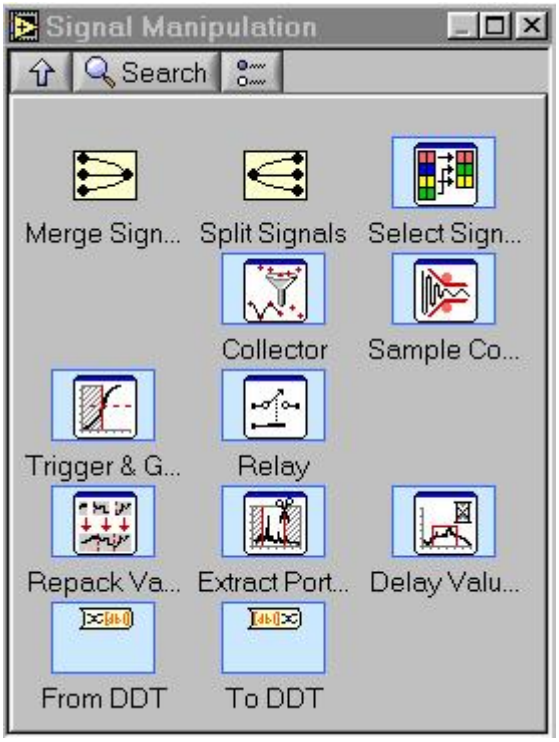
Rys. 15. Paleta funkcji - wyjścia



Rys. 16. Paleta funkcji – struktury sterujące i funkcje czasowe



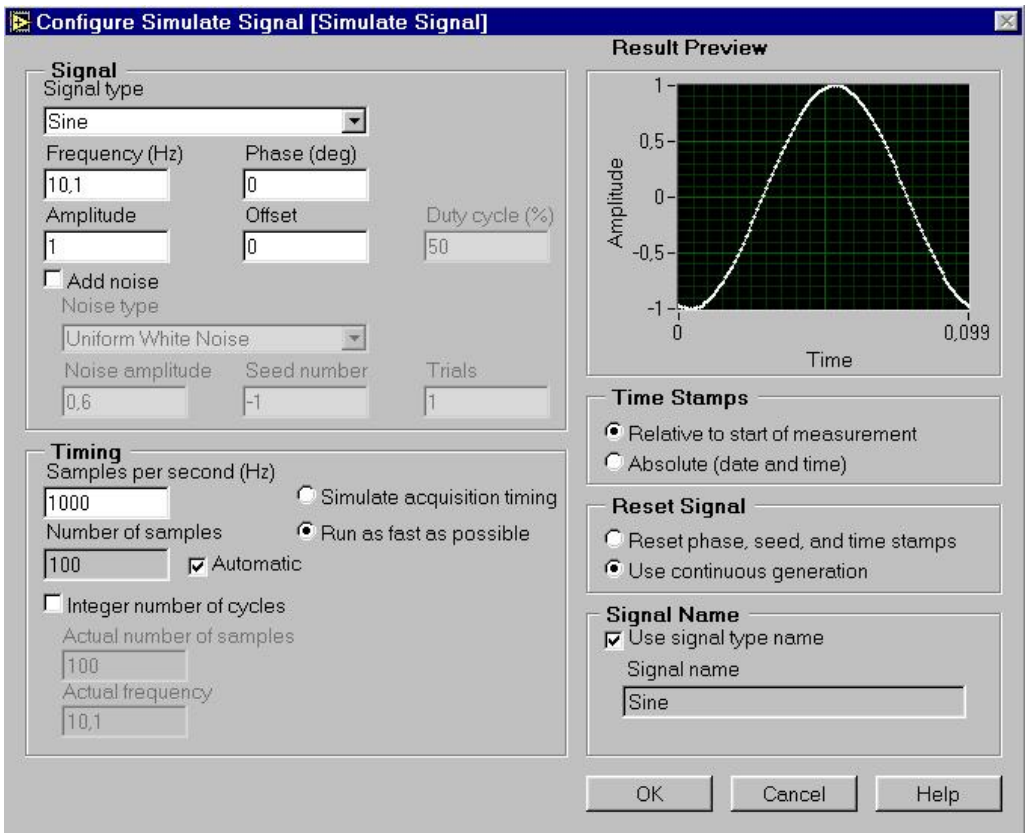
Rys. 17. Paleta funkcji – zależności arytmetyczne i logiczne



Rys. 18 Paleta funkcji – zmiana sygnału

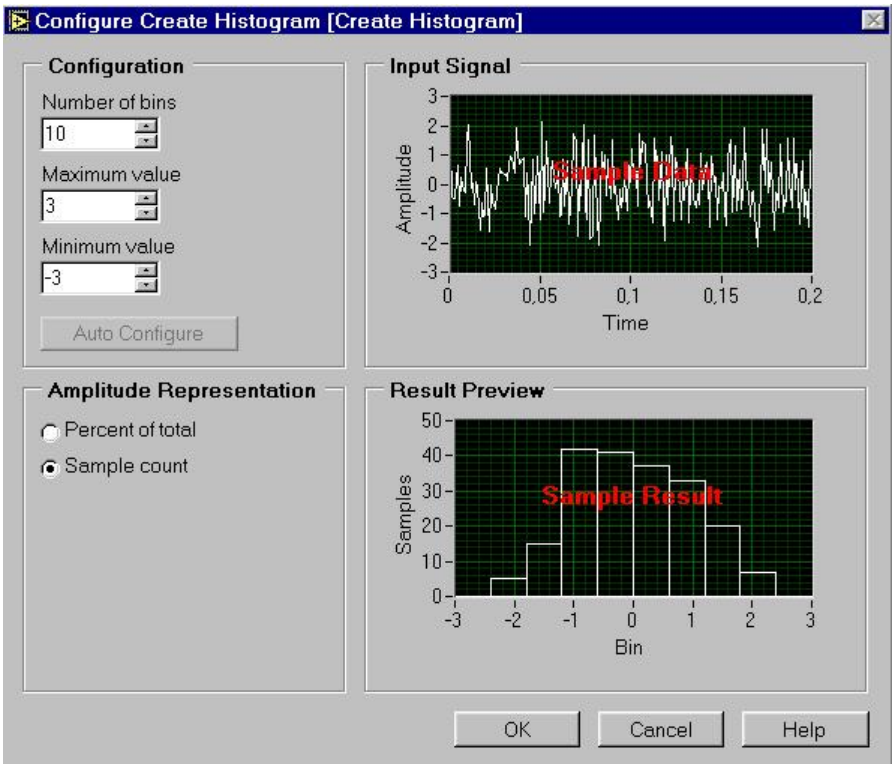
Wybór rodzaju wejść (rys. 13) pozwala na określenie skąd będzie pochodził sygnał, np.:

- rzeczywisty przyrząd pomiarowy dołączony do komputera za pomocą odpowiedniego interfejsu (*Instrument Drivers*),
- sygnał testowy wygenerowany przez program (*Simulate Signal*). Określamy m.in. rodzaj sygnału (np. sinusoidalny, prostokątny, trójkątny, itd.), jego częstotliwość, amplitudę, fazę (rys. 19),
- odczyt danych z pliku (*Read LabView Measurement File*).



Rys. 19. Określanie parametrów sygnału testowego

Sposób analizy sygnałów (rys. 14) pozwala na wygenerowanie sygnału testowego (rys. 19) oraz na otrzymanie histogramu danego przebiegu (*Create Histogram*) (rys. 20).

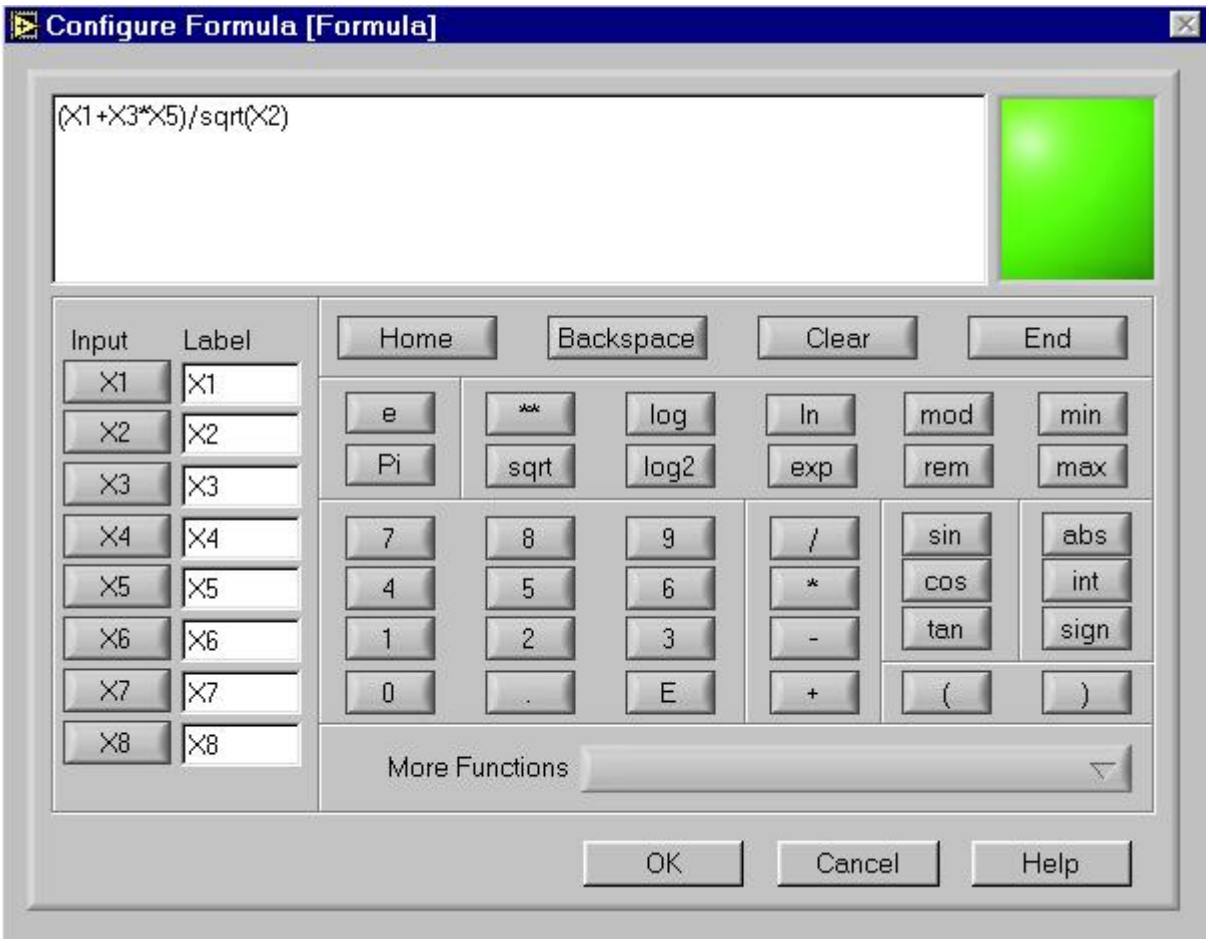


Rys. 20. Określanie parametrów histogramu

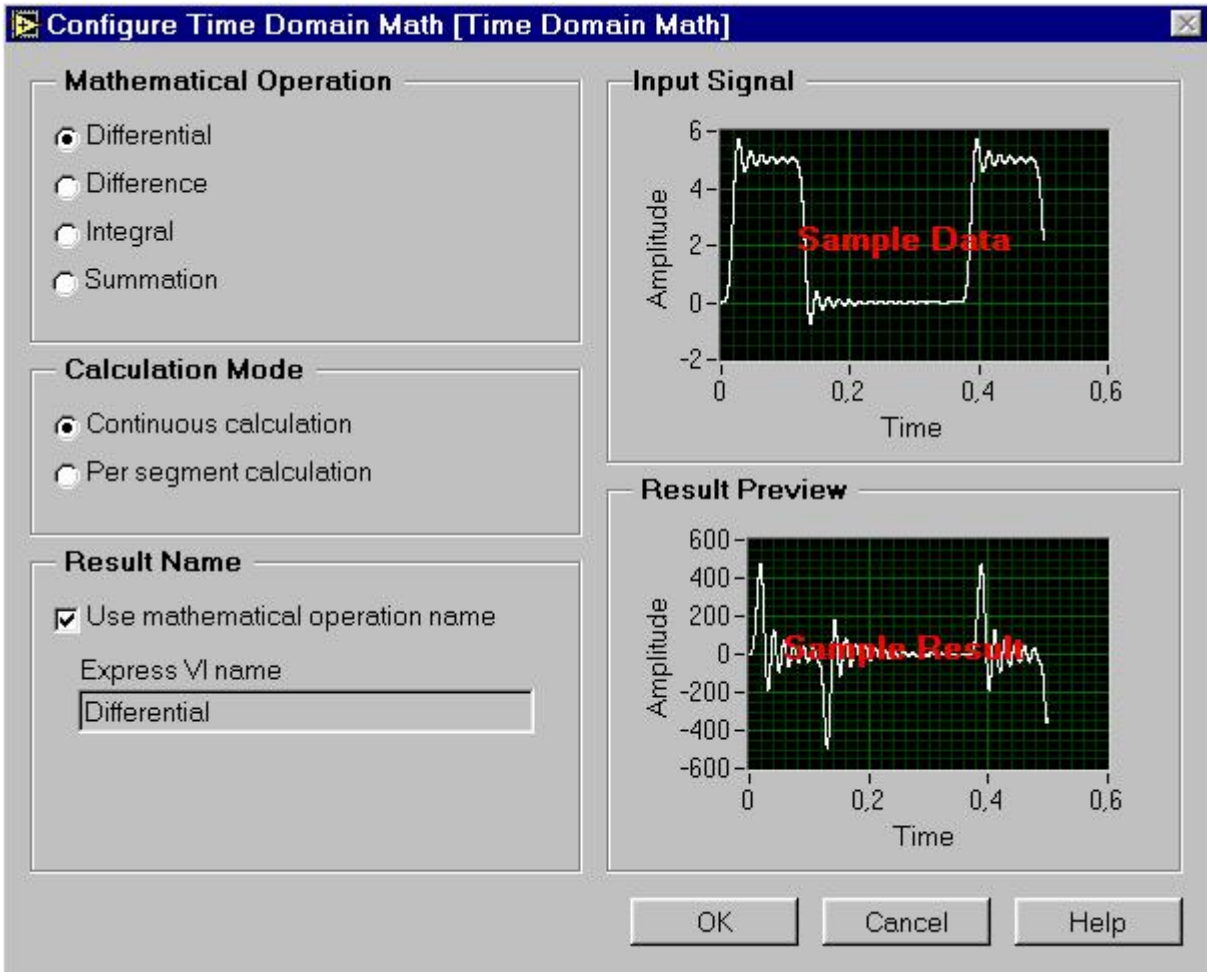
Wybór rodzaju wyjść (rys. 15) pozwala m.in. na:
- zapis otrzymanych wyników do pliku (*Write LabView Measurement File*).
- tworzenie tekstów (*Build Text*),
- tworzenie komunikatów wyświetlanych użytkownikowi (*Display Message to User*).

W programie dostępne są następujące **struktury sterujące i funkcje czasowe** (rys. 16):
- struktura pętli (*While Loop*),
- struktura sekwencji (*Flat Sequence Structure*),
- struktura wyboru (*Case structure*),
- funkcja opóźnienia (*Time Delay*),
- licznik czasu (*Elapsed Time*).

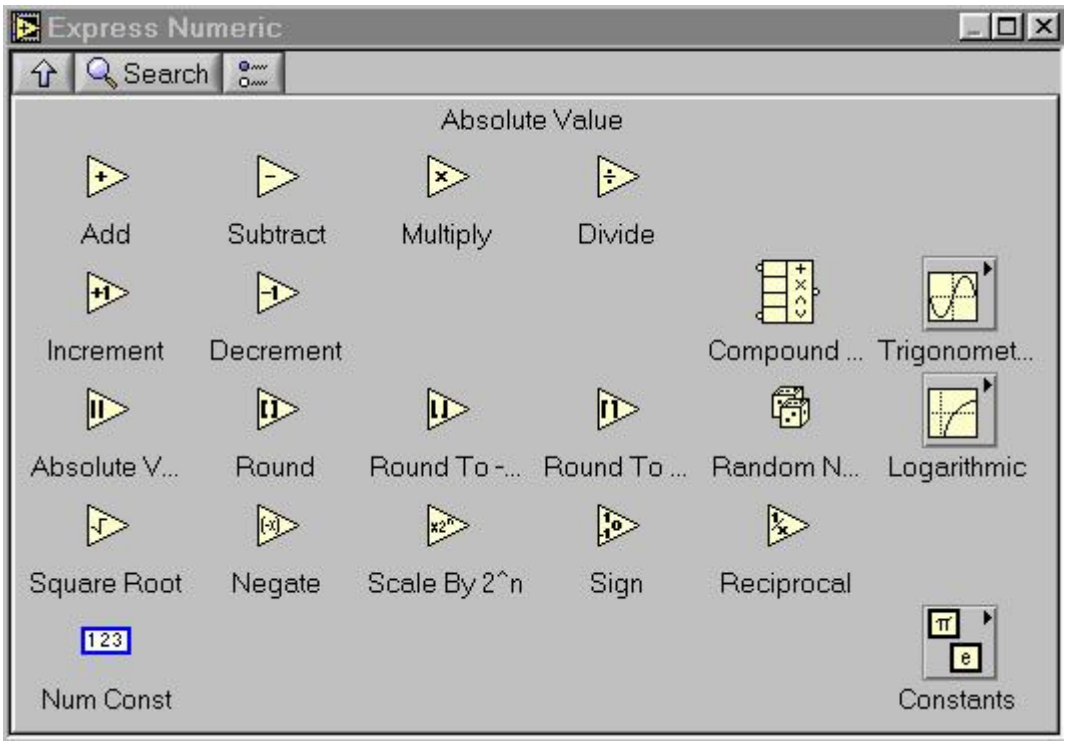
Wśród **zależności arytmetycznych i logicznych** (rys. 17) znajdują się m.in.:
- tworzenie formuł matematycznych (*Formula*) (rys. 21),
- analiza matematyczna sygnału (*Time Domain Math*) (rys. 22 – różniczkowanie *Differential*, różnica *Difference*, całkowanie *Integral*, sumowanie *Summation*),
- zależności arytmetyczne (*Express Numeric*) (rys. 23),
- zależności logiczne (*Express Boolean*) (rys. 24),
- zależności porównawcze (*Express Comparison*) (rys. 25).



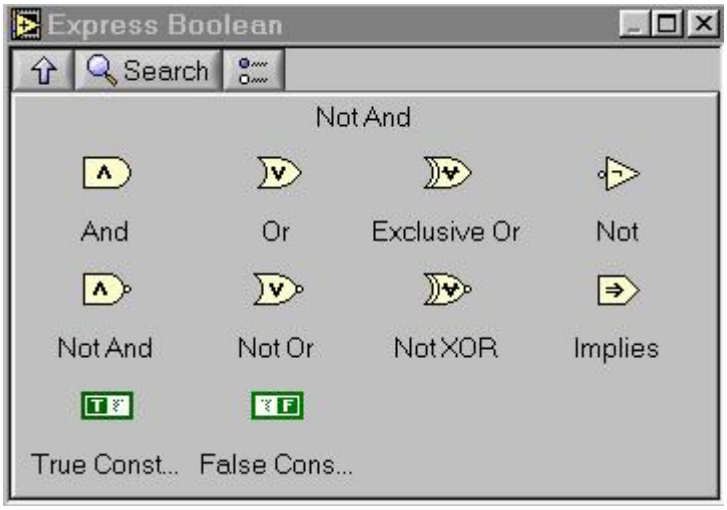
Rys. 21. Tworzenie formuły matematycznej



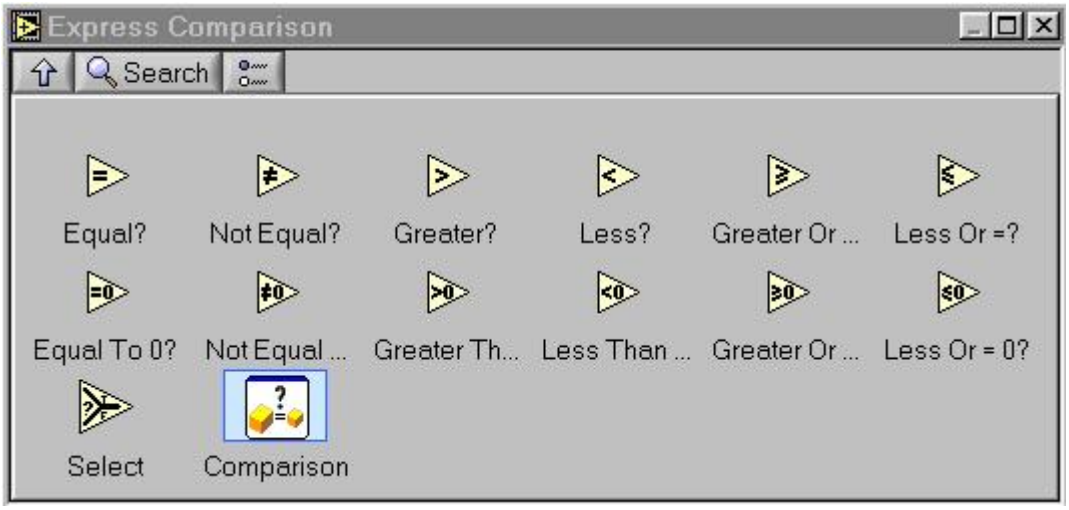
Rys. 22. Okno analizy matematycznej sygnału



Rys. 23. Zależności arytmetyczne

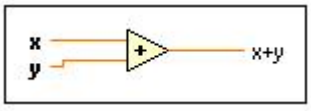
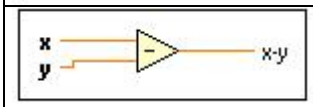
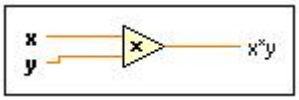
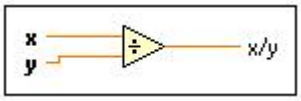
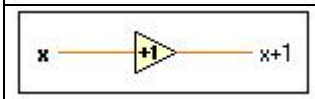
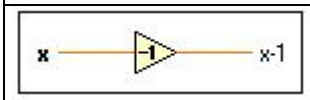


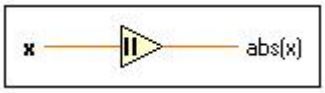
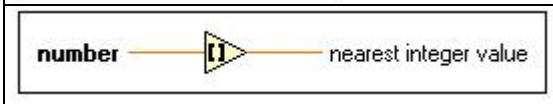
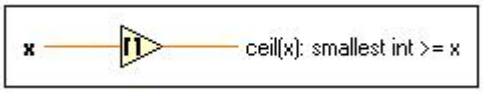
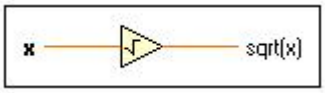
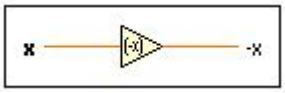
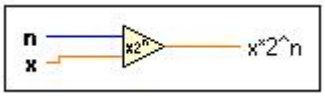
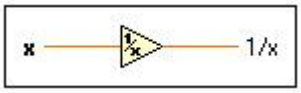
Rys. 24. Zależności logiczne



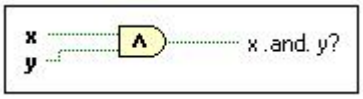
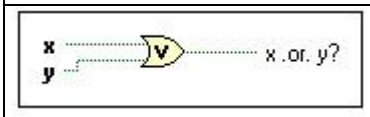
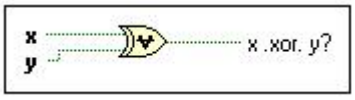
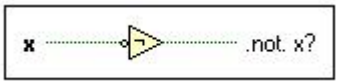
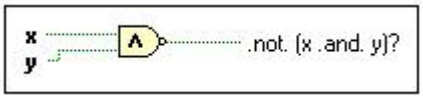
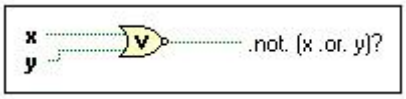
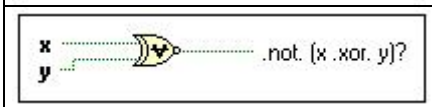
Rys. 25. Zależności porównawcze

Zależności arytmetyczne:

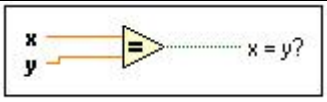
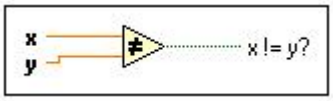
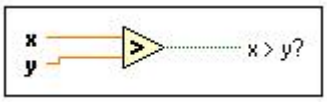
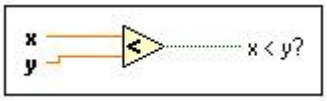
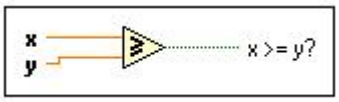
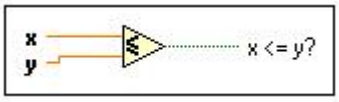
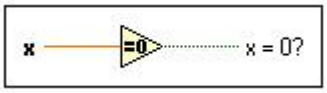
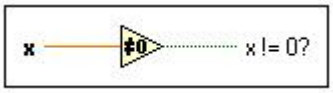
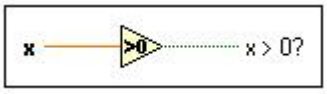
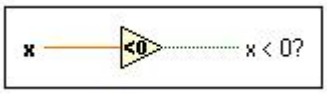
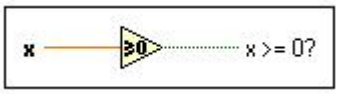
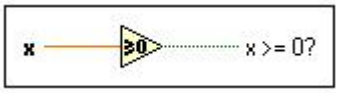
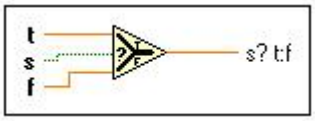
	Suma dwóch wielkości.
	Różnica dwóch wielkości.
	Iloczyn dwóch wielkości.
	Iloraz dwóch wielkości.
	Zwiększa wielkość x o 1.
	Zmniejsza wielkość x o 1

	Wartość bezwzględna wielkości x.
	Zaokrąglenie wielkości x do najbliższej wartości całkowitej.
	Zaokrąglenie wielkości x do najbliższej niższej wartości całkowitej.
	Zaokrąglenie wielkości x do najbliższej wyższej wartości całkowitej.
	Funkcja losowa, która zwraca liczbę zmiennoprzecinkową z przedziału <0,1>
	Pierwiastek kwadratowy wielkości x.
	Negacja wielkości x.
	Mnoży wielkość x przez 2 w potędze n.
	Znak wielkości x.
	Odwrotność wielkości x.

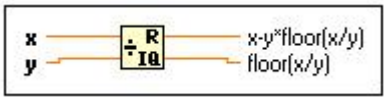
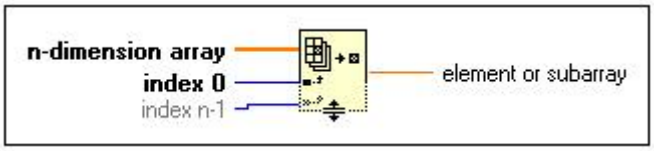
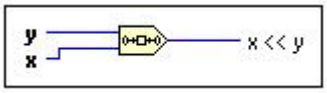
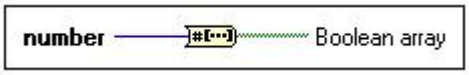
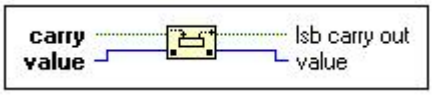
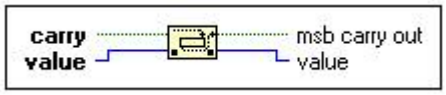
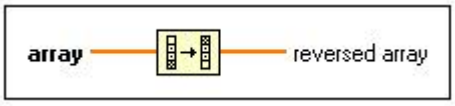
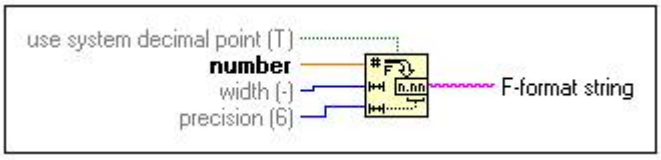
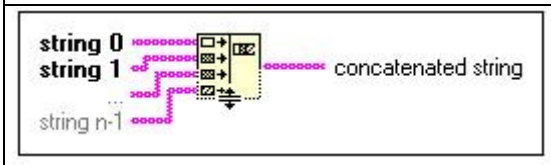
Zależności logiczne:

	Bramka logiczna AND.
	Bramka logiczna OR.
	Bramka logiczna XOR.
	Bramka logiczna NOT.
	Bramka logiczna NAND.
	Bramka logiczna NOR.
	Bramka logiczna XNOR.

Zależności porównawcze:

	Jeśli wielkość x = y, to wynikiem jest wartość 1 (TRUE).
	Jeśli wielkość x ≠ y, to wynikiem jest wartość 1 (TRUE).
	Jeśli wielkość x > y, to wynikiem jest wartość 1 (TRUE).
	Jeśli wielkość x < y, to wynikiem jest wartość 1 (TRUE).
	Jeśli wielkość x ≥ y, to wynikiem jest wartość 1 (TRUE).
	Jeśli wielkość x ≤ y, to wynikiem jest wartość 1 (TRUE).
	Jeśli wielkość x = 0, to wynikiem jest wartość 1 (TRUE).
	Jeśli wielkość x ≠ 0, to wynikiem jest wartość 1 (TRUE).
	Jeśli wielkość x > 0, to wynikiem jest wartość 1 (TRUE).
	Jeśli wielkość x < 0, to wynikiem jest wartość 1 (TRUE).
	Jeśli wielkość x ≥ 0, to wynikiem jest wartość 1 (TRUE).
	Jeśli wielkość x ≤ 0, to wynikiem jest wartość 1 (TRUE).
	Jeśli s jest TRUE, to wynikiem jest wielkość t. Jeśli s jest FALSE, to wynikiem jest wielkość f.

Inne zależności wykorzystywane w stanowisku LV100:

	x-y*floor(x/y) – pozostała „reszta” liczby x, która nie dzieli się całkowicie floor(x/y) - liczba całkowita ilorazu x/y
	Zwraca element tablicy, który odpowiada numerowi indeksu.
	Jeśli y<0, to rejestr przesuwa wartość x (w postaci binarnej) w prawo o y bitów, wstawiając w miejsce bitów o największej wadze 0.
	Zamienia liczbę całkowitą na liczbę binarną.
	Przesuwa w prawo o jeden bit wartość wejściową (*value*). W miejsce najstarszego bitu wstawia bit *carry*.
	Przesuwa w lewo o jeden bit wartość wejściową (*value*). W miejsce najmłodszego bitu wstawia bit *carry*.
	Zwraca wektor, w którym kolejność elementów jest odwrotnością wektora wejściowego.
	Formatuje liczbę wejściową w liczbę o określonej ilości wszystkich cyfr (*width*) i określonej liczbie miejsc po przecinku (*precision*)
	Łączy wszystkie wejściowe łańcuchy znaków w pojedynczy wyjściowy łańcuch znaków.

4. Struktury sterujące

W programie dostępne są następujące struktury sterujące:

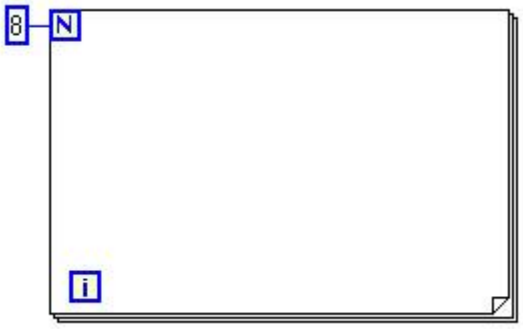
- struktura pętli (*While Loop*),
- struktura sekwencji (*Flat Sequence Structure*),
- struktura wyboru (*Case structure*).

Struktura pętli (*for loop, while loop*)

Stosuje się ją do cyklicznego wykonywania fragmentu programu. Wyróżnia się dwa rodzaje pętli:

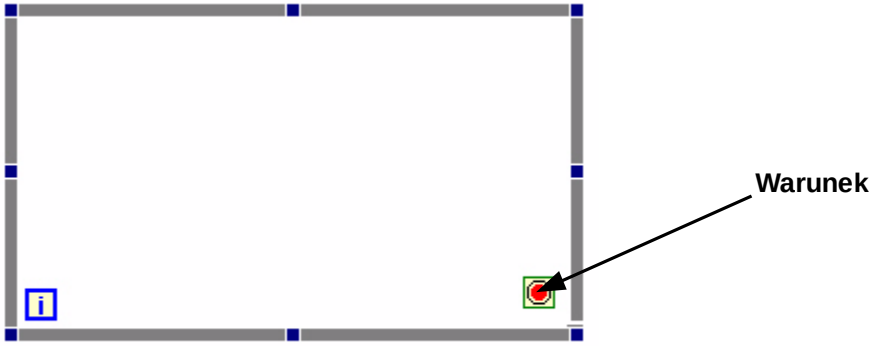
- *for loop*,
- *while loop*.

Pętla *for* przedstawiona jest na rys. 26. W środku obramowania umieszcza się program, który ma być wykonywany N razy (na rys. 26 przyjęto, że N=8, czyli program będzie wykonany 8 razy). Wynika z tego, że musi być znana liczba powtórzeń. Litera „i” jest wyjściem licznika iteracji.



Rys. 26. Pętla for

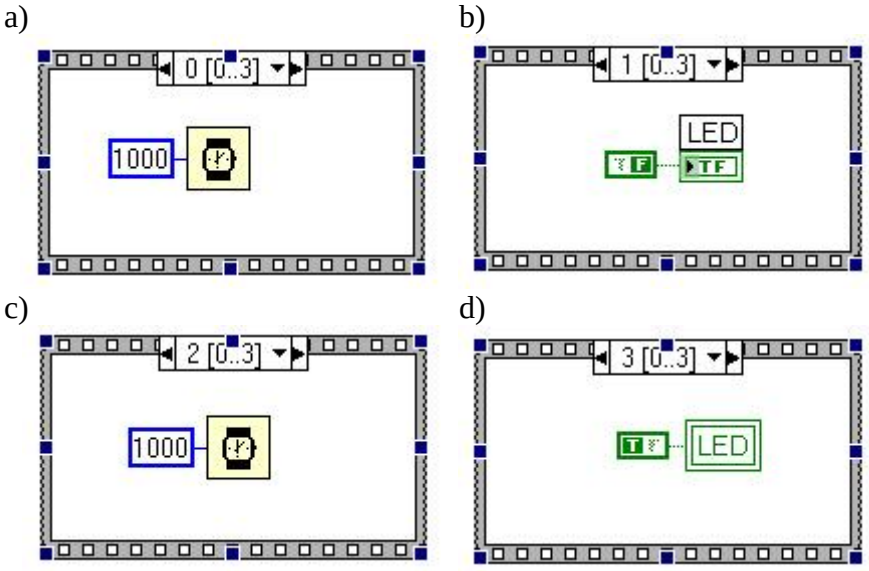
Pętla *while* przedstawiona jest na rys. 27. W środku obramowania umieszcza się program, który wymaga powtórzeń, ale nie jest znana ich liczba. Jest on wykonywany dopóki wartość logiczna podana na wejście „Warunek" jest odpowiednia (TRUE lub FALSE). Warunek ten sprawdzany jest po zakończeniu wykonywania pętli, tak więc pętla zostanie wykonana przynajmniej 1 raz. Litera „i" jest wyjściem licznika iteracji.



Rys. 27. Pętla while

Struktura sekwencji (*Flat Sequence Structure*)

Stosuje się ją do wykonywania kolejnych fragmentów programu, których działanie musi być przeprowadzone w ściśle określonej kolejności. Struktura ta jest przedstawiona na rys. 28.



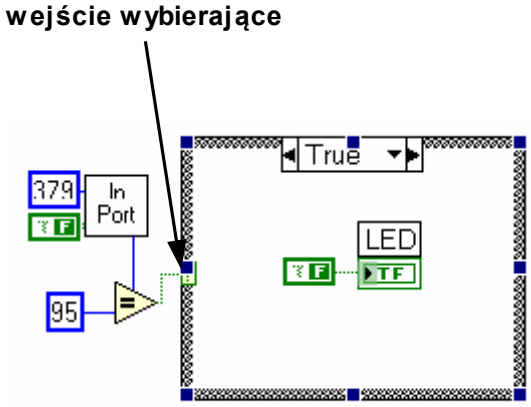
Rys. 28. Struktura sekwencji

Wyglądem struktura sekwencji przypomina ramki filmu, które wykonywane są w kolejności ich numeracji 0, 1, 2, 3, itd. Przykładowo działanie programu przedstawionego na rys. 28 będzie następujące:

- ramka 0: program „czeka” 1 sek. (rys. 28a),
- ramka 1: wyłączenie diody LED (rys. 28b),
- ramka 2: program „czeka” 1 sek. (rys. 28c),
- ramka 3: włączenie diody LED (rys. 28d).

Struktura wyboru (*Case structure*)

Stosuje się ją, gdy zachodzi konieczność alternatywnego wykonywania określonych fragmentów programu. Przedstawiona jest ona na rys. 29. Musi mieć minimum dwie ramki (np. True i False), które wykonywane są w zależności od stanu wejścia wybierającego.



Rys. 28. Struktura wyboru

Wejście wybierające może być m. in. typu boolowskiego (rys. 28) lub całkowitego. W drugim przypadku istnieją następujące możliwości zdefiniowania wartości wybierającej dla poszczególnych ramek:

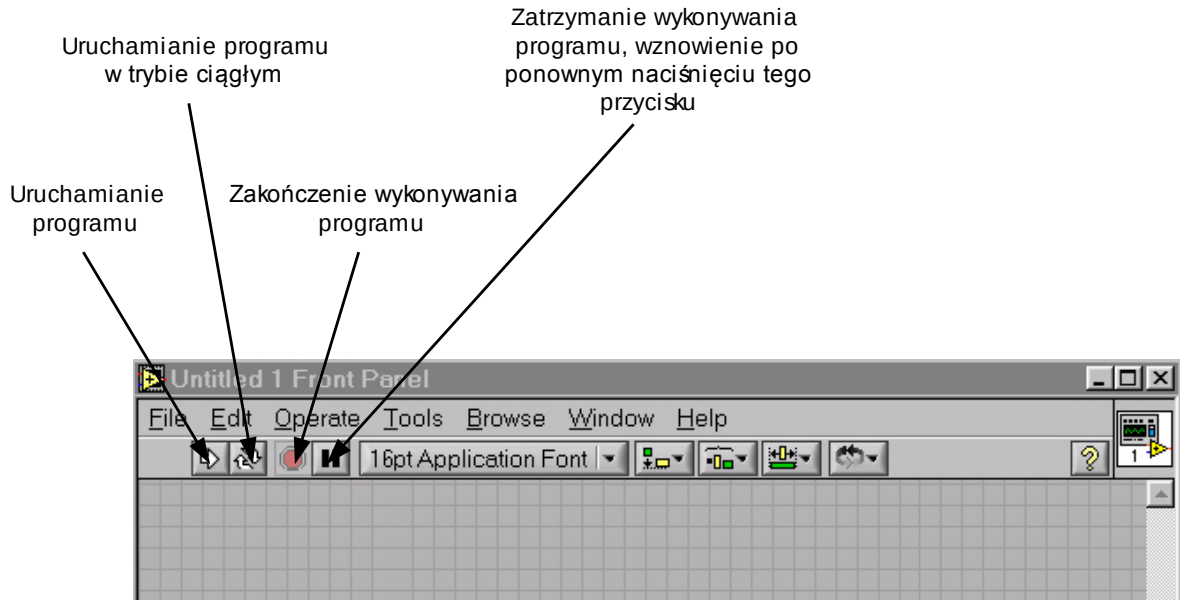
- pojedyncza wartość całkowita (np. 4),
- zbiór wartości całkowitych (np. 2, 4, 5, 12),
- przedział wartości całkowitych (np. 4..8),
- wartość domyślna (*Default*) (obejmuje ona wszystkie pozostałe przypadki wartości wybierającej nie uwzględnione w pozostałych ramkach).

Warunkiem poprawnego napisania programu jest konieczność zdefiniowania w zbiorze ramek wszystkich możliwych przypadków jakie mogą wystąpić w wartości wybierającej.

5. Znaczenie ikon w oknach Panelu Frontowego i Schematu Blokowego

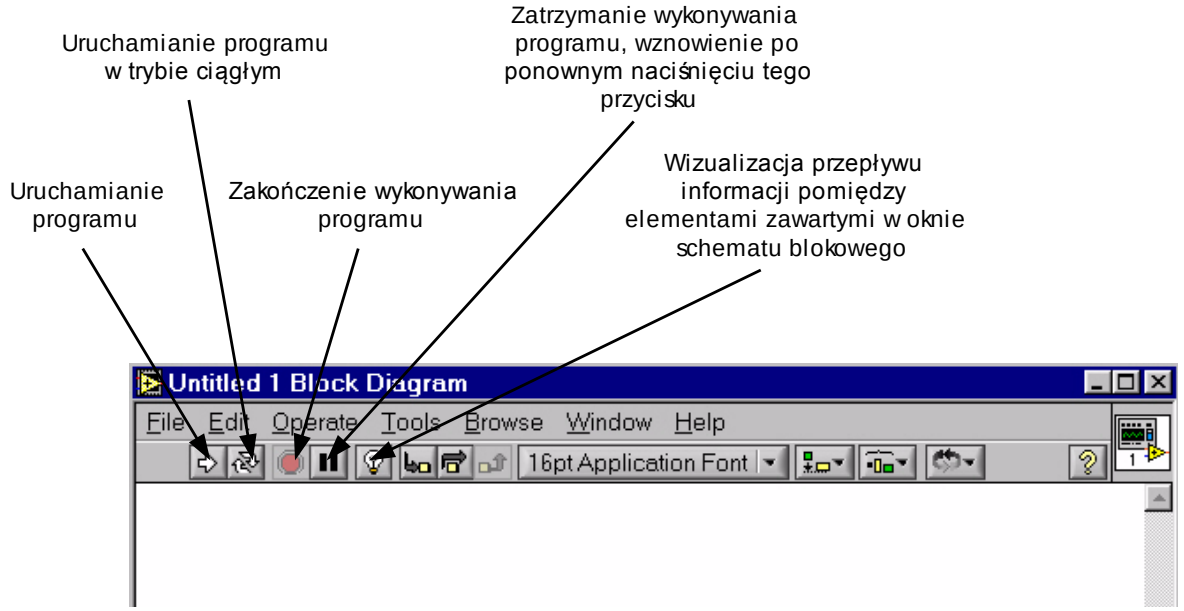
Na rys. 29 przedstawiono znaczenie ikon w oknie Panelu Frontowego. Umożliwiają one uruchomienie zaprojektowanego programu w dwóch trybach:

- praca w trybie „zwykłym” (program będzie działał tak jak został rzeczywiście zaprojektowany),
- praca w trybie ciągłym (program będzie działał cyklicznie aż do momentu zatrzymania przyciskiem „Zakończenie działania programu”).



Rys. 29. Znaczenie ikon – okno panelu frontowego

Na rys. 30 przedstawiono znaczenie ikon w oknie Schematu Blokowego. Zawiera on te same ikony, co okno Panelu Frontowego (rys. 29) oraz dodatkowo ma przycisk umożliwiający wizualizację kolejnych etapów wykonywania programu pomiędzy elementami zawartymi w tym oknie. Jest to szczególnie przydatne przy analizie nowego programu lub szukaniu błędów w tworzonym programie.



Rys. 30. Znaczenie ikon – okno schematu blokowego

6. Wykonywanie kodu programu

Kolejność wykonywania poszczególnych etapów programu określona jest przepływem danych (*data flow*). W przypadku programu o strukturze szeregowej sprawa jest oczywista, ponieważ dalsza część programu będzie wykonana dopiero po „zadziałaniu" wcześniejszych elementów. W przypadku układów bardziej rozbudowanych (struktura równoległa lub mieszana) kolejny fragment programu wykonuje swoją operacje dopiero po uzyskaniu wszystkich danych wejściowych. Dane wyjściowe (uzyskane po wykonaniu operacji)

pojawiają się jednocześnie na wszystkich wyjściach i są jednocześnie przesyłane do następnych bloków programu.

Program może składać się także z wielu niezależnych struktur, które nie muszą być ze sobą powiązane. Kolejność wykonywania poszczególnych etapów ustalana jest przez program LabView automatycznie, przy czym stosuje się tu technikę przeplotu. Gwarantuje to wykonywanie działań w strukturach przemiennie, co można nazwać równoległą realizacją programu.

7. **Literatura**

1. *LabVIEW. User Manual*. National Instruments, kwiecień 2003.