Ćwiczenie

Programowanie w środowisku LabVIEW - Symulator generatora i analizatora przebiegów okresowych.

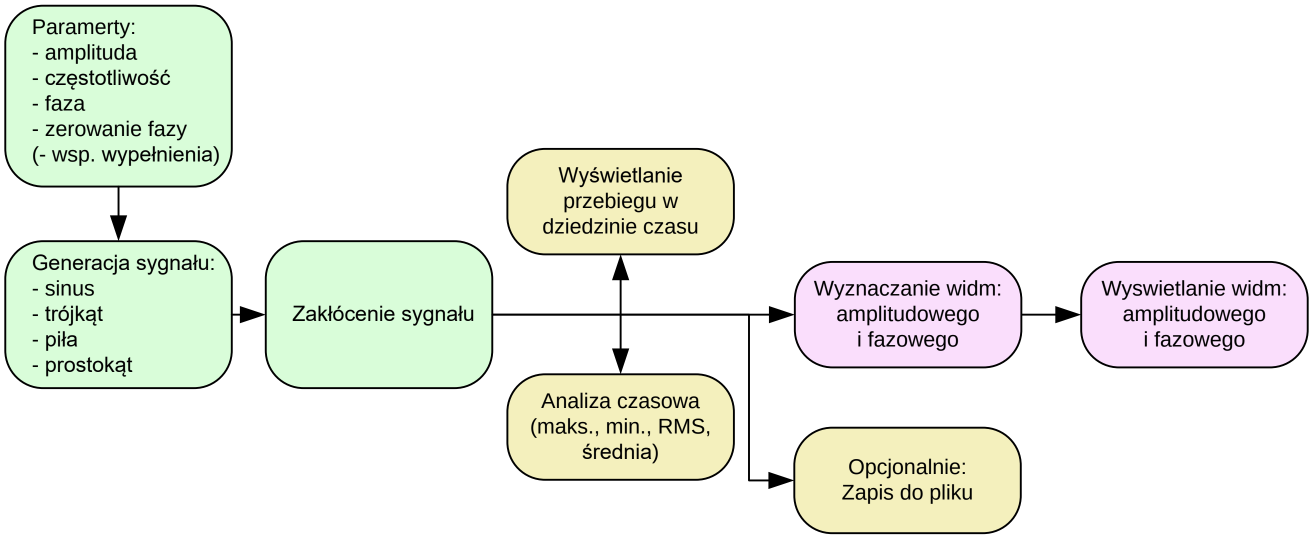
Celem ćwiczenia jest zapoznanie studenta z metodyką programowania, projektowania i tworzenia aplikacji, których zadaniem jest przybliżenie zagadnień i problemów występujących w systemach pomiarowych. W trakcie ćwiczenia student nabędzie podstawowe informacje dotyczące środowiska i umiejętności posługiwania się nim w procesie tworzenia aplikacji pomiarowych.

Zakład Systemów Informacyjno-Pomiarowych

IETiSIP, Wydział Elektryczny, PW

1 Cel i zakres ćwiczenia

Celem ćwiczenia jest zaprojektowanie i realizacja symulatora generatora sygnałów okresowych z możliwością przeprowadzenia analizy częstotliwościowej. Generator powinien charakteryzować się możliwością wyboru wielu różnych przebiegów okresowych i określenia ich: amplitudy, częstotliwości, fazy, liczby próbek i w przypadku przebiegu prostokątnego współczynnika wypełnienia. Schemat blokowy i przykładową płytę czołową przedstawiają rysunki 1a i 1b.



Rysunek 1a. Schemat blokowy symulatora wirtualnego generatora sygnałów okresowych.

Do generacji przebiegów okresowych należy wykorzystać funkcje:

- sinusoidalnego (**SineWave.vi**),
- prostokątnego (**Square Wave.vi**),
- piłokształtnego (**Sawtooth Wave.vi**),
- trójkątnego (**Triangle Wave.vi**).

Generator powinien charakteryzować się możliwością nałożenia szumu (np.: **Uniform White Noise.vi**) i składowej stałej na generowane przebiegi okresowe. Powyższe funkcje można odnaleźć w oknie Functions -> Signal Processing -> Signal Generation.

Zakresy zmian parametrów generowanych przebiegów można przyjąć następująco:

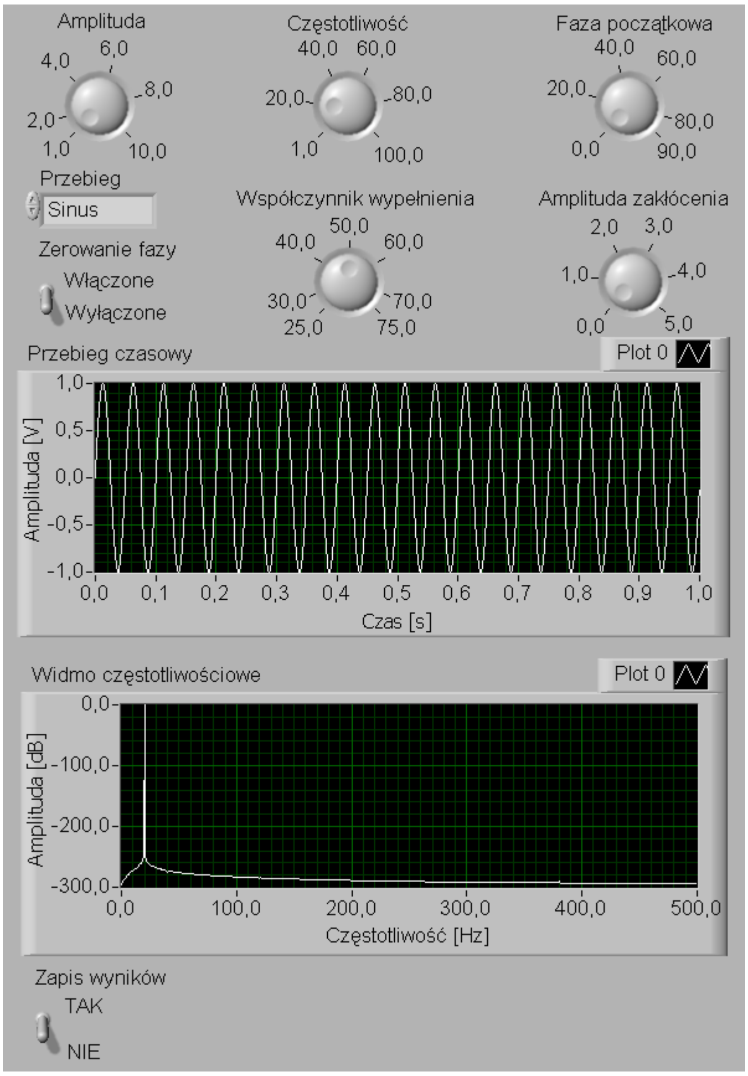
- zmiana amplitudy generowanego przebiegu w zakresie od 1 do 10 V,
- zmiana częstotliwości generowanego przebiegu od 1 do 100 Hz,
- zmiana fazy generowanego przebiegu od 0 do 90 stopni,
- zmian współczynnika wypełnienia dla przebiegu prostokątnego od 25 do 75 procent,
- zmiana amplitudy szumu zakłócającego od 0 to 1 V.

Podane zakresy poszczególnych nastaw ustalone zostały na przykładowym panelu czołowym (Rysunek 1b). **Liczba próbek** może być ustalana z płyty czołowej lub też może to być wartość stała (np: 1000). Parametr **f** (częstotliwość) funkcji z grupy **Wave** podaje się w postaci unormowanej odniesionej do częstotliwości próbkowania. Częstotliwość tę należy przyjąć równą 1000 Hz.

Ponadto generator powinien zostać wyposażony w możliwość podglądu podstawowych parametrów czasowych, tj. wartości:

- maksymalnej i minimalnej (Functions -> Programming -> Array -> **Array Max & Min.vi**),
- średniej (Functions -> Programming-> Mathematics -> Probability and Statistics -> **Mean.vi**),

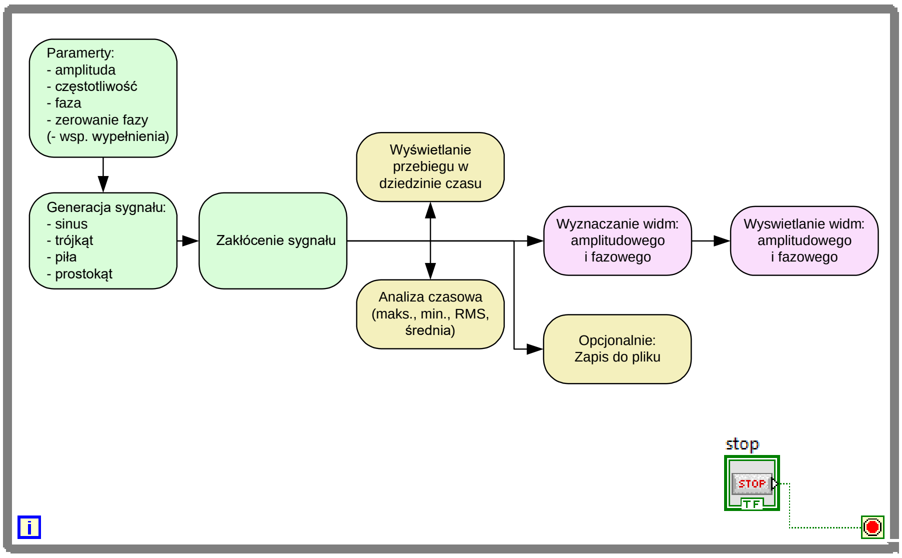
- skutecznej (Functions Programming -> Mathematics -> Probability and Statistics -> **RMS.vi**).



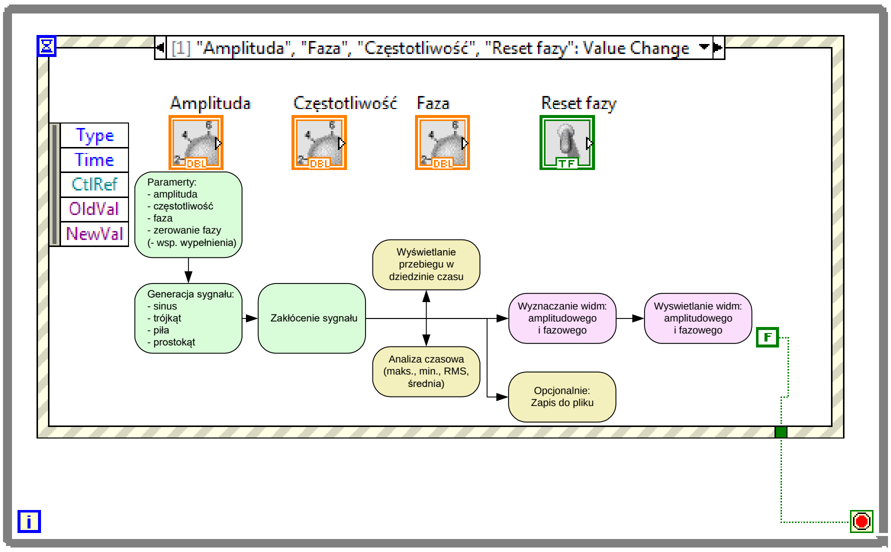
Rysunek 1b. Przykładowa płyta czołowa symulatora wirtualnego generatora sygnałów okresowych

2 Realizacja zadania.

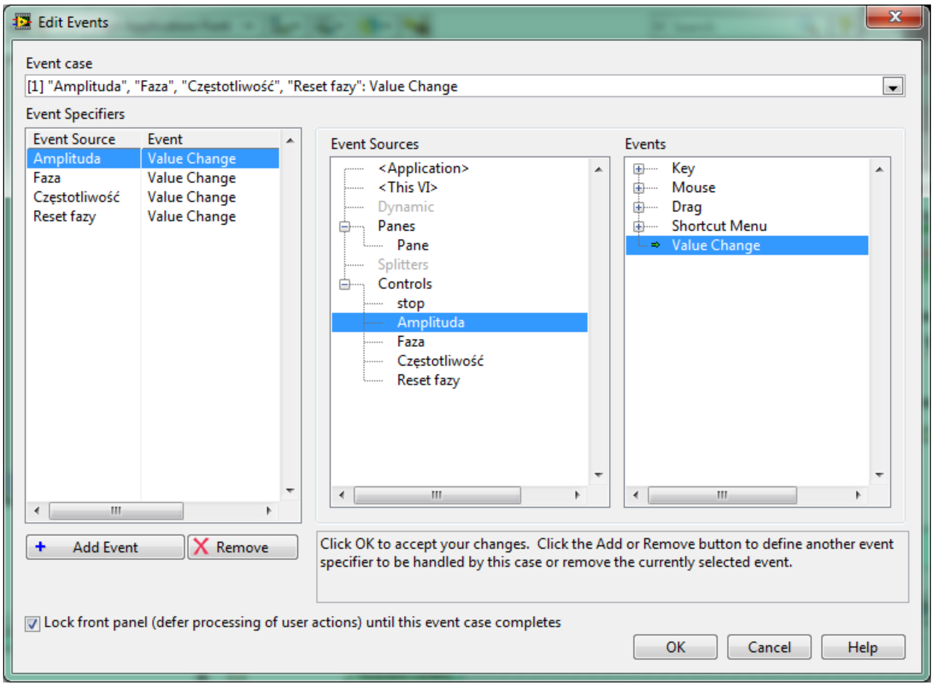
Projektowany generator powinien posiadać możliwość prezentacji widma częstotliwościowego wybranego przebiegu. Preferowaną skalą jest skala logarytmiczna; opcjonalnie generator może mieć możliwość wyboru pomiędzy skalą liniową i logarytmiczną. Należy pamiętać o odpowiednim przeskalowaniu osi czasu i częstotliwości odpowiednich wykresów. Wyznaczanie transformaty Fouriera należy zrealizować w postaci funkcji (subVI). W najprostszej wersji kod realizujący zadanie mógłby być realizowany bez żadnych dodatkowych instrukcji. Wtedy pojedyncze uruchomienie programu  powodować będzie pojedynczą reakcję programu. Można temu zaradzić uruchamiając program w sposób ciągły . Lepszym rozwiązaniem jest umieszczenie całego kodu programu wewnątrz pętli **while** (rys. 1c) i wykorzystanie przycisku  do kontroli zakończenia programu. Przycisk **STOP** ma domyślnie ustawiony tryb pracy **Latch When Released** (Patrz Dodatek A), co zapewni jego prawidłową pracę w pętli. Wariant programu z pętlą **while** ma jednak tę niedogodność, że funkcje wewnątrz pętli wykonywane są niezależnie od tego czy użytkownik dokonuje zmian w nastawach na płycie czołowej czy też nie. W wyniku mamy do czynienia z pewnym zbędnym obciążeniem zasobów komputera. Rozwiązaniem tego problemu jest zastosowanie struktury **Event Structure** wewnątrz pętli **while**. W tym przypadku kod programu wykonywany będzie jedynie w przypadku zmiany jednej z nastaw. W najprostszym podejściu wystarczy utworzyć osobną zakładkę **Event Structure** dla wszystkich nastaw na płycie czołowej za wyjątkiem przycisku **STOP**, tj. dla amplitudy i amplitudy zakłóceń, częstotliwości, fazy i resetowania fazy, rodzaju przebiegu oraz współczynnika wypełnienia dla prostokąta. Dla wszystkich nastaw można wybrać zdarzenie **Value Change**, zgodnie z rysunkiem 1e. Zakładka **Timeout** może pozostać pusta, gdyż domyślnie Timeout jest wyłączony (parametr -1 na wejściu), jednakże dobrze jest zamieścić wewnątrz tej zakładki stałą logiczną FALSE (lub zależnie od wersji LabVIEW) i połączyć ją (poprzez automatycznie utworzony tunel) z terminalem zakończenia pętli **while** . Identycznie należy postąpić w przypadku zakładki dla nastaw. Ona również powinna zawierać stałą FALSE połączoną z terminalem końca pętli **while**. Poza dwoma omówionymi zakładkami **Event structure** powinna zawierać zakładkę dla obsługi przycisku STOP (**Value Change**). W tym przypadku terminal przycisku STOP powinien znajdować się wewnątrz zakładki i analogicznie jak dla zakładek pozostałych powinien zostać połączony z terminalem końca pętli **while**.



Rysunek 1c. Realizacja zadania z wykorzystaniem pętli **while**.

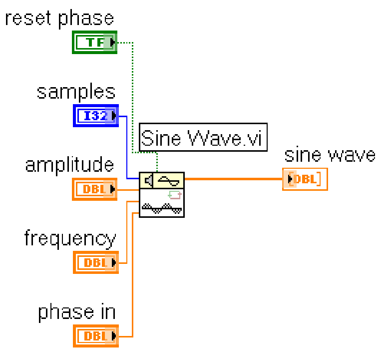


Rysunek 1d. Realizacja zadania z wykorzystaniem pętli **while** oraz **Event structure**

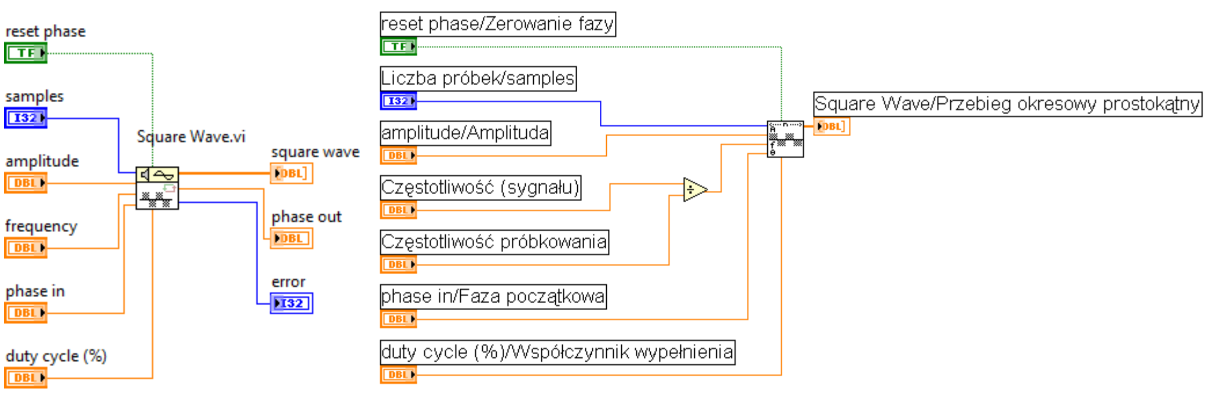


Rysunek 1e. Definiowanie zdarzeń dla poszczególnych nastaw z płyty czołowej.

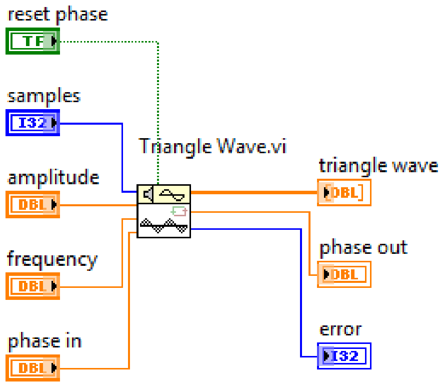
W tym miejscu w LabVIEW może być też dodatkowo pomocne wykorzystanie zdarzenia **Mouse Up** w miejsce **Value Change** w **Event Structure**.
Wejścia i wyjścia funkcji z grupy **Wave**, z których należy skorzystać w projektowane j aplikacji przedstawiają rysunki od 2 do 5.



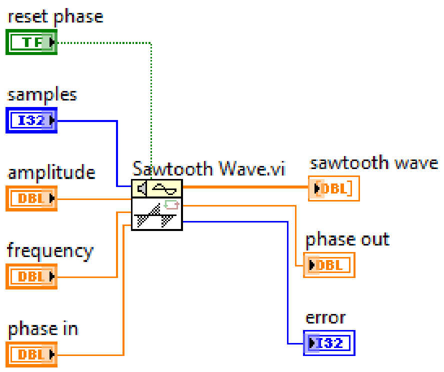
Rysunek 2.Wejścia i wyjścia dla funkcji **Sine Wave**.



Rysunek 3.Wejścia i wyjścia dla funkcji **Square Wave**.

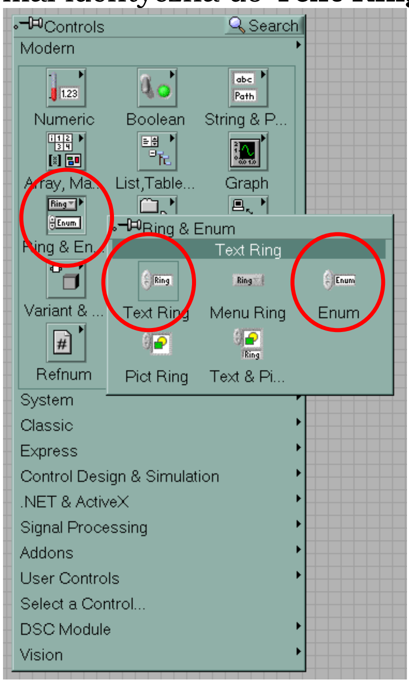


Rysunek 4. Wejścia i wyjścia dla funkcji **Triangle Wave**.



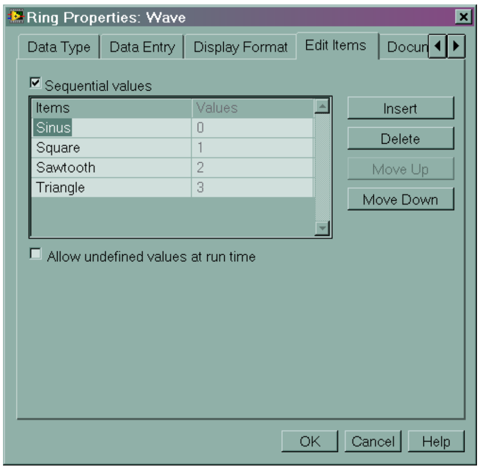
Rysunek 5. Wejścia i wyjścia dla funkcji **Sawtooth Wave**.

Jak widać poszczególne wejścia i wyjścia dla wymienionych wyżej funkcji mają takie samo znaczenie. Jedynym wyjątkiem jest funkcja **Square Wave**. Posiada ona dodatkowe wejście dla ustalenia współczynnika wypełnienia **Duty Cycle**. Do wyboru generowanego przebiegu odpowiednia będzie konstrukcja **Case** (Functions -> Programming - > Structures -> Case Structure). Jako przełącznika dla wyboru generowanego sygnału można użyć elementu **Text Ring** (Controls -> Modern -> Rings&Enum -> Text Ring – rysunek 6) lub **Enum** (rysunek 6). Konfiguracja tego drugiego jest niemal identyczna do **Text Ring** i ma kilka dodatkowych zalet.

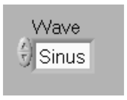


Rysunek 6. “Przełącznik” **Text Ring** w oknie Controls.

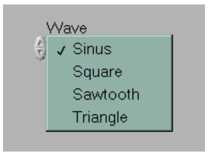
Jeżeli element **Text Ring** jest już wybrany, należy ustawić jego własności. W tym celu należy otworzyć okno **Properties**. Najprostszym sposobem jest otwarcie menu kontekstowego dla elementu **Text Ring** (prawym klawiszem myszy na elemencie **Text Ring**). Następnie należy wybrać zakładkę **Edit items**. Symulator powinien pracować poprawnie dla czterech różnych przebiegów , zatem elementy przełącznika **Text Ring** muszą odpowiadać poszczególnym przebiegom (Rysunek 7).



Rysunek 7. Okno własności przełącznika **Text Ring**.

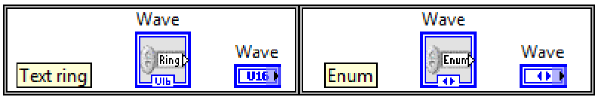


Rysunek 8. Wygląd przełącznika **Text Ring** na płycie czołowej.

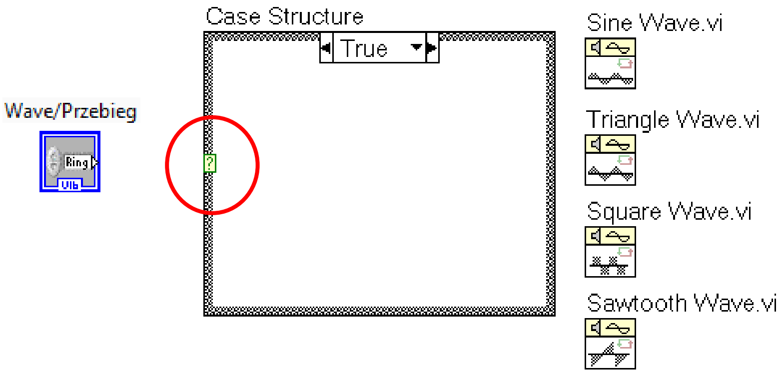


Rysunek 9. Menu kontekstowe dla skonfigurowanego przełącznika **Text Ring** do wyboru przebiegu.

W przykładzie opisany został element **Text Ring**. Widok jego terminala zależy od lokalnej konfiguracji. Alternatywnie można użyć elementu typu **Enum**. Terminale zależnie od wyboru i konfiguracji mogą wyglądać jak poniżej.

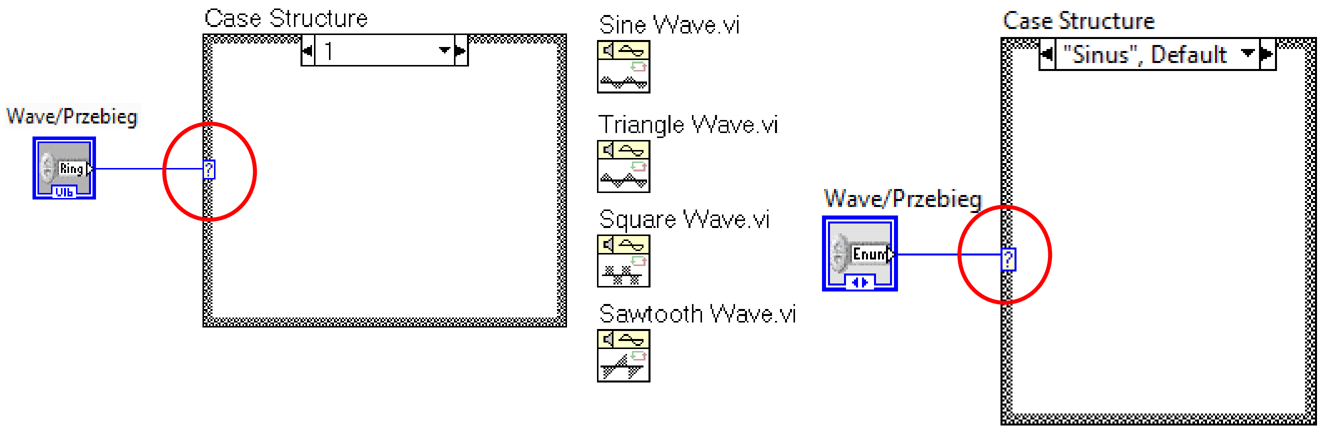


Każdą pozycję, w elemencie typu **Ring** reprezentuje pewna określona wartość całkowita, stanowiąca indeks (numer) tej pozycji. Pozycje indeksowane są począwszy od zera. Zatem po stronie diagramu odpowiedni terminal przyjmować może wartości wynikające z ustawień przełącznika znajdującego się na płycie czołowej. Za pomocą tego terminala (przypisanego do przełącznika wyboru przebiegu) i konstrukcji **Case** można sterować generowaniem tego przebiegu, który został wybrany. Domyślnie (rysunek 10) konstrukcja **Case** ma możliwość obsługi warunków logicznych (0 lub 1, *true* lub *false*).



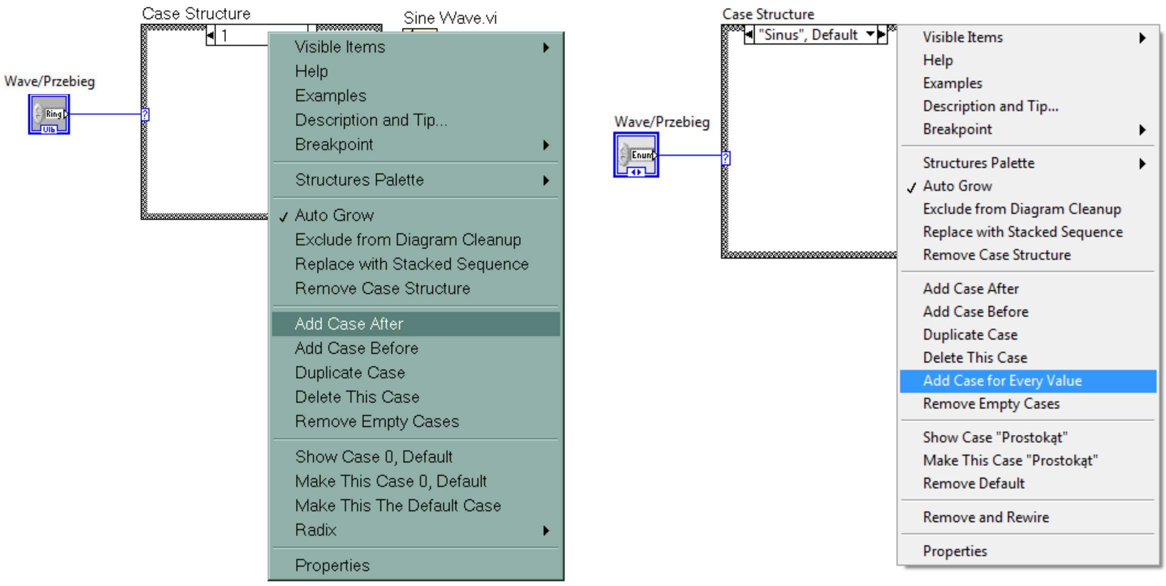
Rysunek. 10. Struktura **Case**.

Jeżeli jednak podłączy się do selektora warunku element, którego wartości są typu całkowitoliczbowego (skonfigurowany uprzednio przełącznik do wyboru przebiegu) konstrukcja automatycznie zmieni tryb działania (rysunek 11). Należy tylko, poprzez menu kontekstowe dodać dodatkowe pozycje (**Case**), aby móc obsłużyć wszystkie przypadki zdefiniowane w przełączniku.



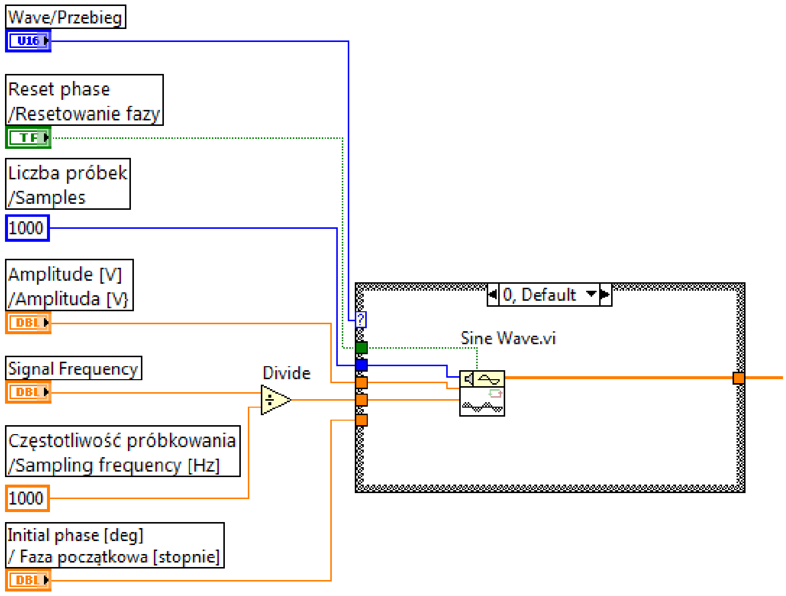
Rysunek 11. Zmodyfikowana struktura **Case**. W przypadku **Text Ring** w **Case Structure** pojawiają się pozycje przełącznika **Wave/Przebieg**. W przypadku **Enum** w **Case Structure** pojawiają się **nazwy** pozycji przełącznika **Wave/Przebieg**.

Domyślnie struktura **Case** zawiera jedynie dwa okna: jedno dla wartości logicznej 0 i drugie odpowiednio dla wartości 1. Aby dodać kolejne (w programie powinno być ich łącznie cztery, po jednym dla każdego przebiegu), należy skorzystać z menu kontekstowego struktury **Case**. W tym celu należy na wybrać kliknąć prawym klawiszem myszy na okienku zakładek struktury **Case** (górna część obramowania tej struktury) i wybrać pozycję **Add Case After** dwa razy (Rysunek 12). W przypadku jeśli użyjemy przełącznika typu **Enum** otrzymamy dodatkowo możliwość **Add Case for Every Value** (Rysunek 12).



Rysunek 12. Menu kontekstowe dla struktury **Case**. **Text Ring**-strona lewa, **Enum**-strona prawa.

Następnie należy umieścić poszczególne funkcje do generacji przebiegów zgodnie z porządkiem przedstawionym na rysunku 7. Wszytki terminale nastaw (rysunki od 2 do 5) powinny być umieszczone poza strukturą **Case**. NA rysunku 13 zamieszczony został przykład dla funkcji **Sine Wave**. Należy w tym miejscu pamiętać, że funkcja **Square Wave** ma dodatkowy parametr **Duty Cycle**.



Rysunek 13. Parametry wejściowe funkcji **Sine Wave**.

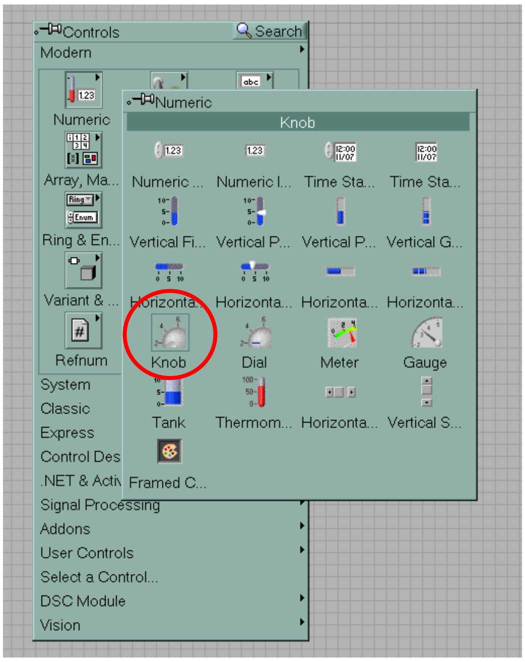
W tym miejscu raz jeszcze należy przypomnieć, że częstotliwość sygnału podaje się jak wartość unormowaną względem częstotliwości próbkowania. Jest to wymagane przez wszystkie funkcje z grupy **Wave**. Stąd wartość częstotliwości sygnału (**Frequency [Hz]** na rysunku 13) jest podzielona przez częstotliwość próbkowania (stałą **Sampling Frequency** na rysunku 13).

Poszczególne parametry generowanych przebiegów tj.:

- Phase in [deg] / Faza początkowa [stopnie],
- Amplitude [V] / Amplituda [V],

- Signal Frequency [Hz] / Częstotliwość próbkowania [Hz],
- and Duty Cycle [%] / Współczynnik wypełnienia [%]

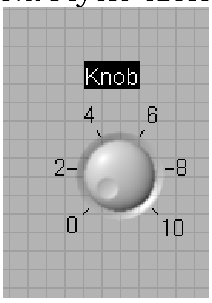
mogą być zadawane za pomocą przełącznika typu **Knob** (Controls -> Numeric -> Knob - Rysunek 14).



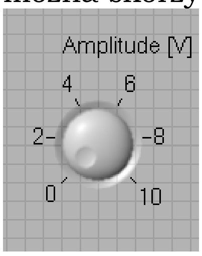
Rysunek 14. Przełącznik **Knob** w oknie **Controls**.

Jako przykład konfiguracji, poniżej podano sposób postępowania dla nastawy amplitudy. W tym celu należy wykonać kolejno następujące kroki:

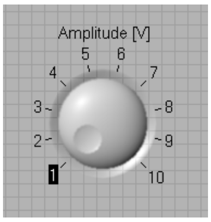
1. Na Płycie czołowej aplikacji – **Front Panel** należy umieścić potencjometr typu **Knob**.



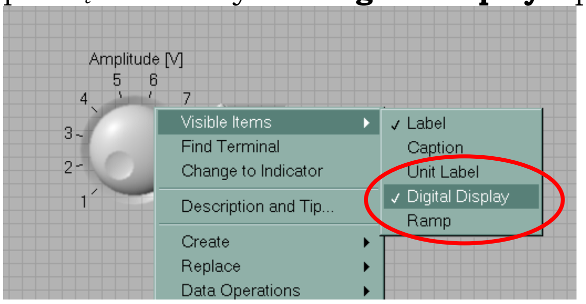
2. Domyślną etykietę/nazwę należy zmienić na Amplituda [V]/Amplitude [V]. W tym celu można skorzystać z narzędzia **Edit text tool** (**Tools window**).



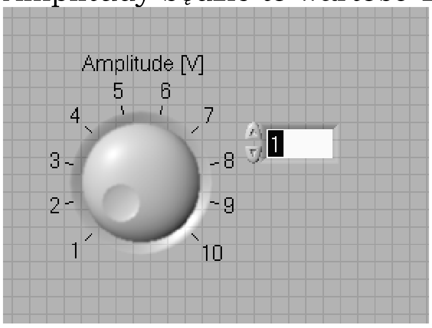
3. Za pomocą narzędzia **Operate Value tool** (**Tools window**) należy zmienić wartość minimalną na 1 a wartość maksymalną na 10 (należy kliknąć odpowiednią wartość i następnie ją edytować). Przedstawiony poniżej przełącznik został dodatkowo powiększony **Position/Size/Select** (**Tools window**) tak aby poszczególne wartości etykiet miały równomierny rozkład.



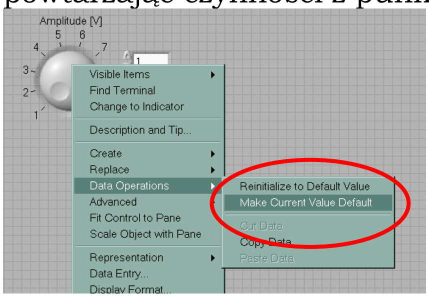
4. Ustawienie wartości domyślnej (W wersji 8.6 czynność ta wydaje się nieco skomplikowana. W wersjach poprzednich ustawienie wartości domyślnej było uproszczone.) W celu ustawienia wartości domyślnej należy otworzyć menu kontekstowe (prawy klawisz myszy) przełącznika i wybrać **Digital display** z podmenu **Visible Item**.



5. Następnie w oknie **Digital display** należy wprowadzić określoną wartość (w przypadku Amplitudy będzie to wartość 1).

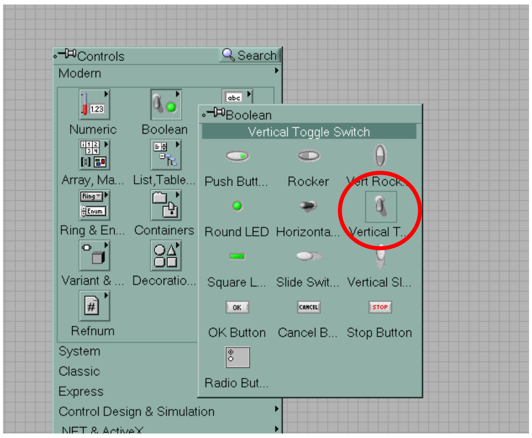


6. Na koniec, ponownie z menu kontekstowego należy wybrać pozycję **Make Current Value Default** z podmenu **Data Operation**. Dodatkowo można wyłączyć dodatkowy wyświetlacz powtarzając czynności z punktu 4, aby go dezaktywować (wyłączyć).



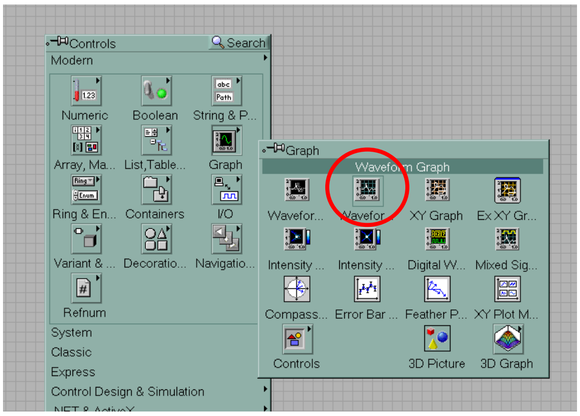
7. Powyższe czynności należy powtórzyć dla pozostałych nastaw, określając odpowiednie wartości dla wartości maksymalnych, minimalnych i domyślnych. (Amplituda : minimum 1, maksimum 10, wartość domyślna 1; Częstotliwość: minimum 1, maksimum 100, wartość domyślna 10; Faza początkowa: minimum 0, maksimum 90, wartość domyślna 0; Współczynnik wypełnienia: minimum 25, maksimum 75, Wartość domyślna 50; Zakłócenie: minimum 0, maksimum 1, wartość domyślna 0.

W celu umożliwienia zerowania fazy początkowej należy zastosować przełącznik binarny **Vertical Toggle Switch** (Controls -> Modern -> Boolean -> Vertical Toggle Switch – rysunek 15).



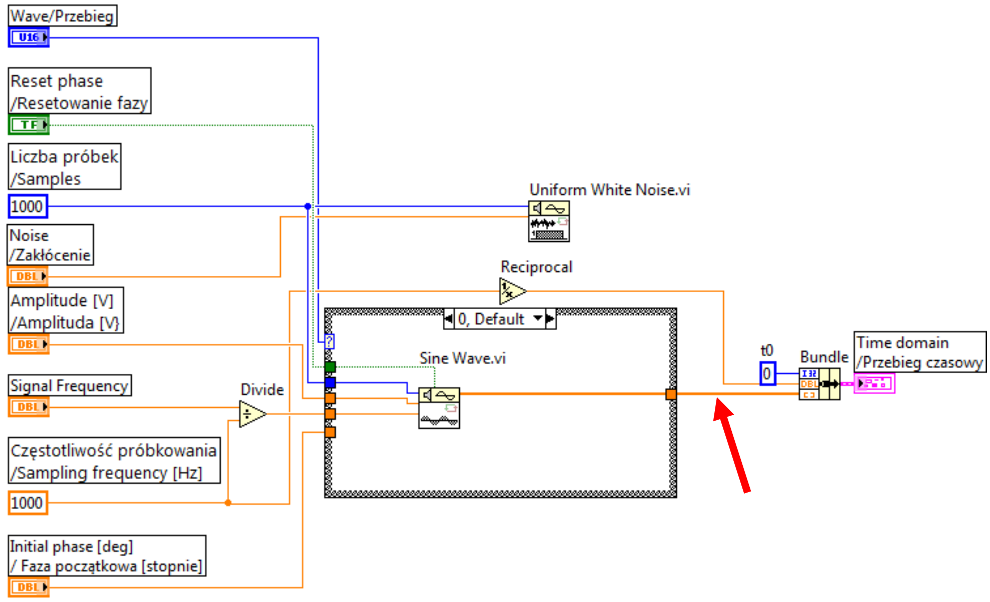
Rysunek 15. Położenie przełącznika **Vertical toggle Switch** w oknie **Controls**.

Prezentację wygenerowanego przebiegu można zrealizować z wykorzystaniem wyświetlacza oscyloskopowego typu **Waveform Graph** (Controls -> Modern - > Graphs -> Waveform Graph – rysunek 16).



Rysunek 16. Wyświetlacz oscyloskopowy **Waveform Graph** w oknie **Controls**.

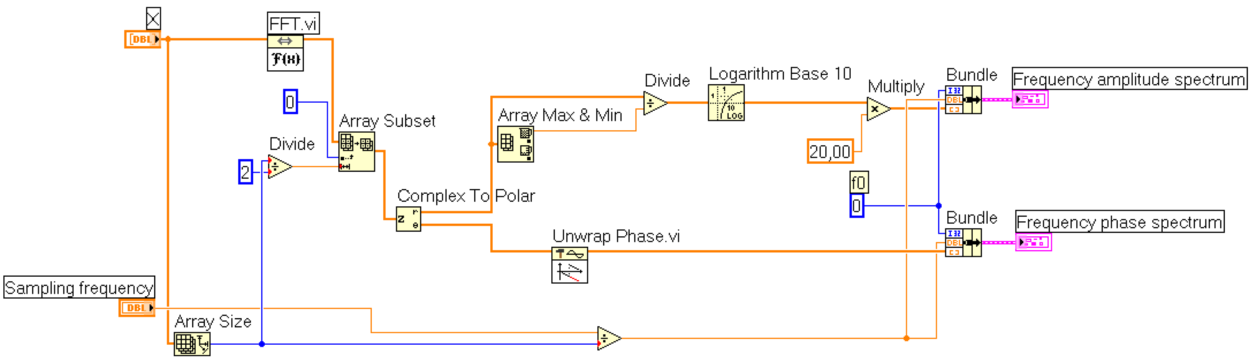
Nazwę wyświetlacza należy zmienić na **Time domain/Dziedzina czasu** (Najprościej zmienić nazwę zaraz po umieszczeniu nowego wyświetlacza na płycie czołowej aplikacji, kiedy to nazwa domyślna jest podświetlona. Czynność tę można również przeprowadzić wykorzystując narzędzie **Edit Text** z okna **Tools**). Wyświetlacz mógłby zostać bezpośrednio podłączony do wyjścia funkcji **Wave** umieszczonych w strukturze **Case** – czerwona strzałka na rysunku 17. Lepszym rozwiązaniem jest jednak wyskalowanie osi X w jednostkach czasu. W tym celu należy dodatkowo wykorzystać funkcję **Bundle** w celu utworzenia klastra **Waveform**. Klaster ten składa się z trzech pól: t_0 (chwilę początkową należy ustawić na wartość 0); okres próbkowania (odwrotność częstotliwości próbkowania); wektor próbek sygnału (wyjście funkcji **Wave**).



Rysunek 17. Przykład wykorzystania funkcji **Bundle**.

Na schemacie z rysunku 17 zastosowano funkcję **Uniform White Noise** przeznaczoną do generacji zakłóceń. Jednakże wynik działania tej funkcji nie został wykorzystany do zniekształcania sygnałów okresowych. Aby zrealizować to zadanie należy przerwać połączenie oznaczone czerwoną strzałką na rysunku 17. Następnie w miejsce usuniętego połączenia należy umieścić funkcję sumatora **Add**. Wejścia funkcji **Add** powinny zostać połączone odpowiednio do wyjść funkcji **Wave** (struktura **Case**) and wyjścia funkcji **Uniform White Noise**. Wyście funkcji **Add** powinno został podłączone do pozostałego wejścia funkcji **Bundle**.

Generowany sygnał powinien być przedstawiany bezpośrednio w dziedzinie czasu oraz po operacji transformacji także w dziedzinie widma częstotliwościowego. W przypadku widma częstotliwościowego wyniki należy przedstawić w skali logarytmicznej. Opcjonalnie rodzaj skali może być ustalany przez użytkownika z poziomu płyty czołowej. Osie czasu i częstotliwości (X) powinny być wyskalowane w odpowiednich jednostkach (**Waveform cluster**). Pożądane jest aby przeprowadzenie obliczeń związanych z transformacją FFT wykonane zostało w dedykowanym subVI. Przykład kodu LabVIEW do realizacji przekształcenia Fouriera przedstawiony został na rysunku 18.



Rysunek 18. Przykład realizacji funkcji wyznaczającej widmo częstotliwościowe.

Wejście X na rysunku 18 jest wejściem sygnału z wyjść funkcji **Wave**, częstotliwość próbkowania (sampling frequency) jest już zdefiniowana w programie głównym. Widma amplitudowe i fazowe mogą zostać przedstawione na wyświetlaczach typu **Waveform Graph** (Controls -> Modern - > Graphs -> Waveform Graph).

3 Realizacja pozostałych funkcji.

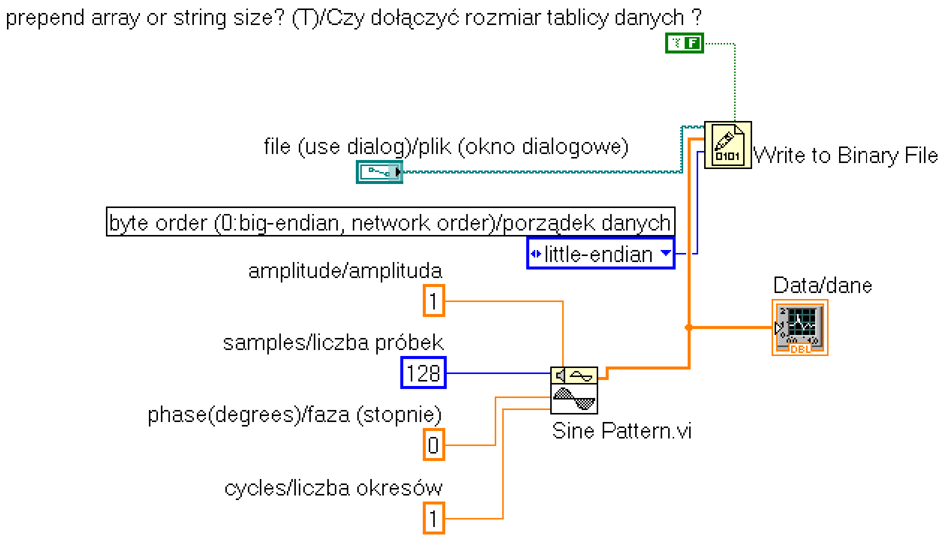
Pozostałe funkcje:

- określenie wartości maksymalnych i minimalnych
    - (LabVIEW8.6: Functions -> Programming -> Array -> **Array Max & Min.vi**),
- określenie wartości średniej
    - (LabVIEW8.6: Functions -> Programming-> Mathematics -> Probability and Statistics -> **Mean.vi**),
- określenie warości skutecznej
    - (LabVIEW8.6: Functions Programming -> Mathematics -> Probability and Statistics -> **RMS.vi**).

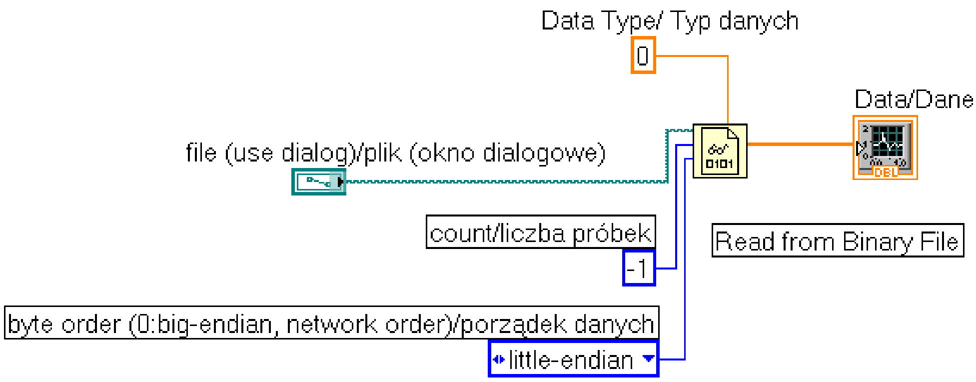
Wyświetlacze Dziedzina czasu/Time domain oraz Frequency spectrum/ Widmo częstotliwościowe są typu **Waveform Graph**. Należy odpowiednio zmienić ich nazwy. (Najprościej zmienić nazwę zaraz po umieszczeniu nowego wyświetlacza na płycie czołowej aplikacji, kiedy to nazwa domyślna jest podświetlona. Czynność tę można również przeprowadzić wykorzystując narzędzie **Edit Text** z okna **Tools**).

Zadania dodatkowe:

1. Zapis i/lub odczyt do i/lub z pliku sygnału generowanego w dziedzinie czasu (operacje na żądanie). W tym celu wskazówką mogą być schematy przedstawione na rysunkach 19 i 20.
2. Zmiana skali widma częstotliwościowego - liniowa/logarytmiczna (modyfikacja funkcji wyznaczającej widmo częstotliwościowe – rysunek 18)
3. Zmiana okna dyskretnego w transformacji Fouriera
4. Wyznaczanie (automatyczne) podstawowej częstotliwości przebiegu - funkcja **Array Max &Min** dla widma. Operacja musi działać prawidłowo dla różnych wartości liczby próbek i częstotliwości próbkowania.
5. Pomiary kursorami (np: delta X, delta Y, automatycznie maksimum lub minimum).
6. Oznaczenie poziomą linią, na wykresie oscyloskopowym wartości np.: maksymalnej, minimalnej, średniej (operacje na żądanie).
7. Dodanie rejestracji sygnału rzeczywistego - funkcja DAQ Assistant.



Rysunek 19. Uproszczona wersja funkcji do zapisu danych.



Rysunek 20. Uproszczona wersja funkcji do odczytu danych(plik czytamy po jego utworzeniu i zapisie) .

Położenie pozostałych przydatnych funkcji:

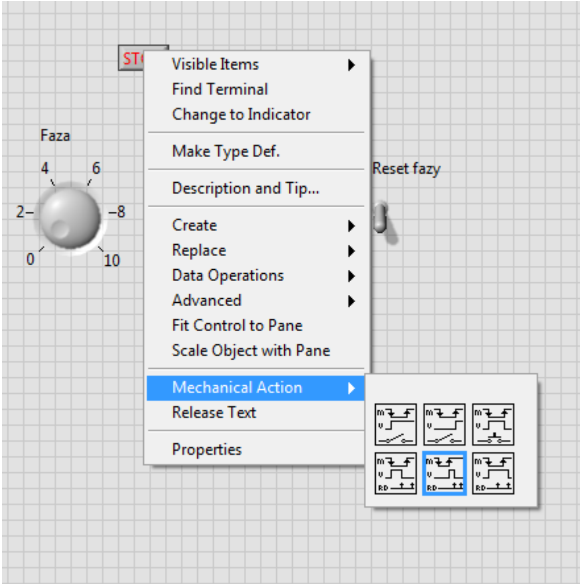
Functions -> Signal Processing -> Signal Operation -> **Unwrap Phase.vi**
Functions -> Signal Processing -> Transform-> **FFT.vi**
Functions -> Programming -> Numeric -> Complex -> **Complex To Polar**
Functions -> Programming -> Cluster -> **Bundle**
Functions -> Mathematics->Elemetary&Special Functions -> Expotentional Functions -> **Logarithm Base 10**
Functions -> Programming -> Array -> **Array Size**
Functions -> Programming -> Array -> **Array Max &Min**

Dodatek A Tryby pracy przycisków

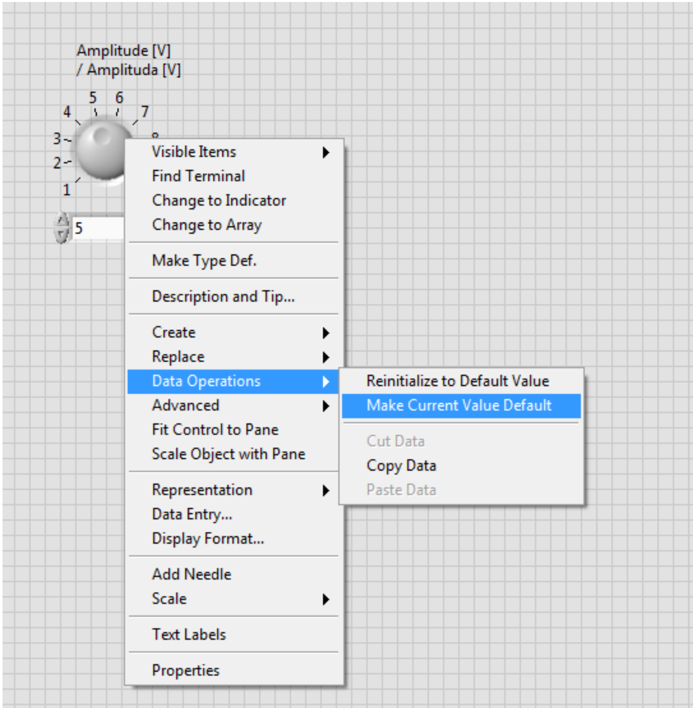
W zależności od wymagań można ustawić różne tryby pracy dla elementów na płycie czołowej. W szczególności dotyczy to przycisków, które mogą pracować w następujących trybach:

- Switch When Pressed
- Switch When Released
- Switch Until Released
- Latch When Pressed
- Latch When Released
- Latch Until Released

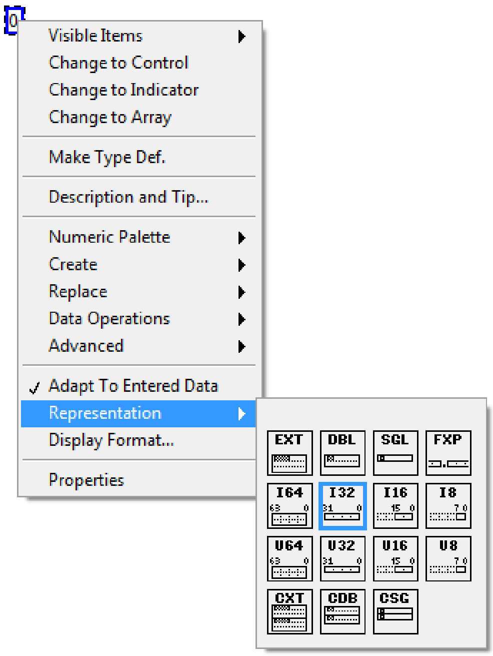
W zadaniu w przypadku przycisku **STOP** dobrze jest ustawić tryb pracy jako **Latch When Released** (rys. A1). Wtedy po naciśnięciu przycisku (przycisk przyjmie wartość logiczną **TRUE**) zostanie on "zatrzaśnięty i zapamiętany" dopóki jego stan nie zostanie odczytany przez program. Po odczytaniu przycisk powraca do stanu "nieaktywnego" czyli **FALSE**. Ten mechanizm jest bardzo wygodny gdyż zapewnia reakcję programu na zdarzenie z płyty czołowej. Łatwo jest bowiem wyobrazić sobie sytuację, w której użytkownik naciska przycisk w momencie gdy przez program wykonywane jest inne zadanie niż odpytywanie stanu przycisku. Co więcej czynności wykonywane w odpowiedzi na naciśnięcie przycisku wykonywane są tylko raz, bez zbędnej redundancji.



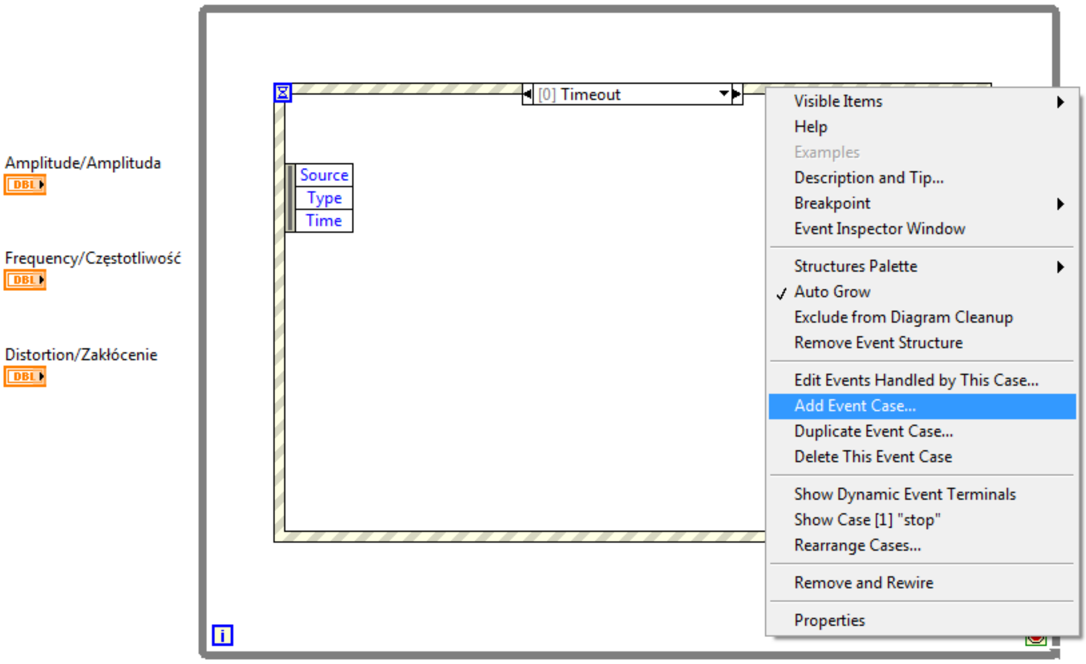
Rysunek A1. Ustawianie trybu pracy dla przycisku **STOP**.



Rysunek A2. Ustawienie własnych wartości domyślnych.



Rysunek A3. Zmiana typu danych dla stałej (również w ten sam sposób można zmienić przypisany typ dla elementów na płycie czołowej).



Rysunek A4. **Event structure** - dodawanie nowego przypadku Event Case (menu kontekstowe- klikamy prawym klawiszem myszy na obramowaniu **Event Structure**).

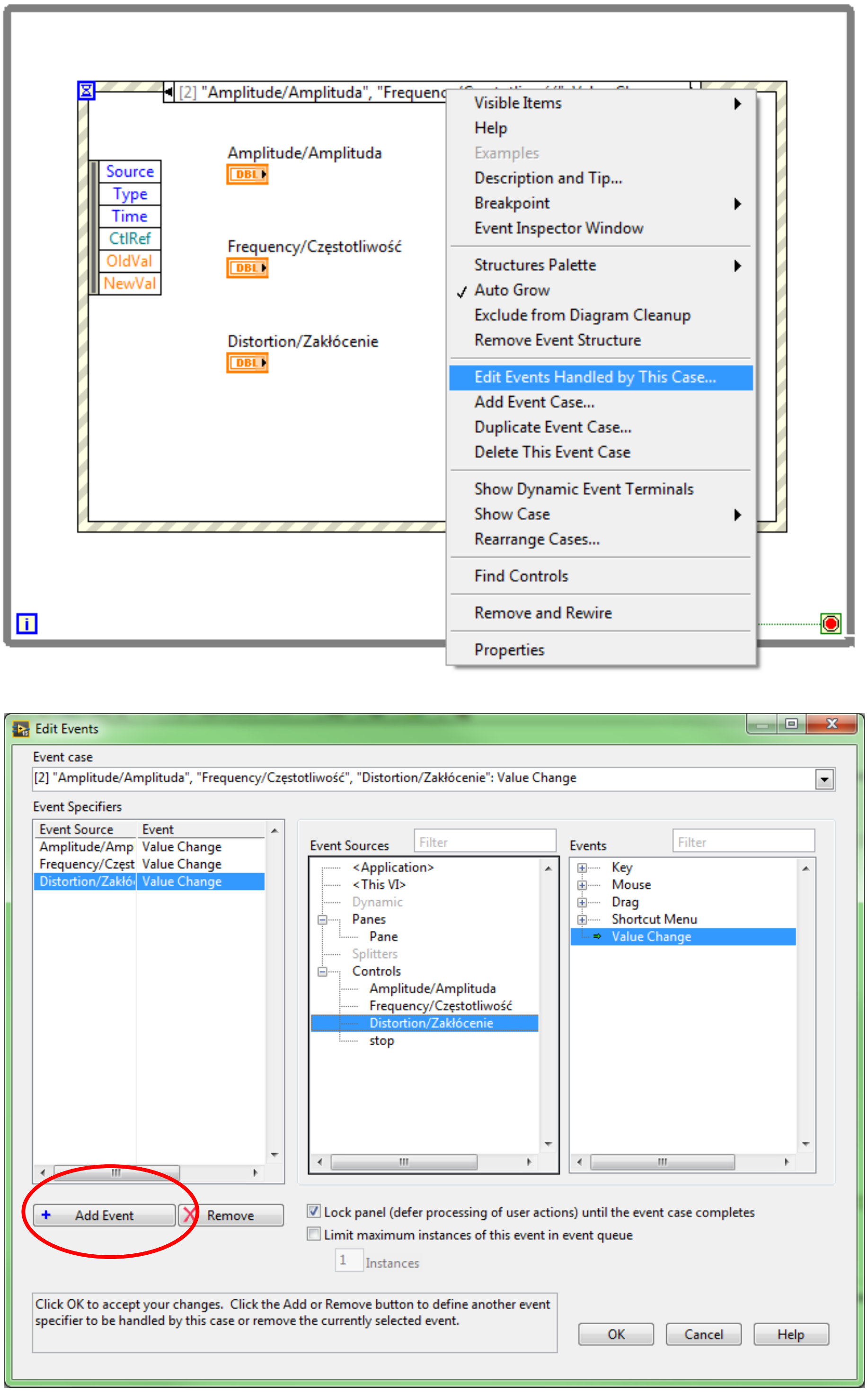


Figure A5. **Event structure** - modyfikacja istniejącego przypadku Event Case (menu kontekstowe- klikamy prawym klawiszem myszy na obramowaniu **Event Structure**).