CYFROWE PRZETWARZANIE SYGNAŁÓW Laboratorium Automatyka i Robotyka, studia stacjonarne pierwszego stopnia	Instytut Metrologii, Elektroniki i Informatyki
Temat: **Analiza widmowa okresowych sygnałów deterministycznych**	Ćwiczenie 2

1. Cel ćwiczenia

Celem ćwiczenia jest praktyczne poznanie zagadnień związanych z analizą widmową deterministycznych sygnałów okresowych poprzez samodzielną implementację w języku ANSI C z wykorzystaniem środowiska LabWindows/CVI firmy National Instruments programu wyznaczającego widmo amplitudowe i fazowe na podstawie próbek sygnału oraz przeanalizowanie wpływu uzupełniania zerami i stosowania okien czasowych na wyznaczane widma.

2. Zagadnienia do przygotowania

- Moduł i argument liczby zespolonej.
- Próbkowanie sygnałów ciągłych – twierdzenie o próbkowaniu.
- Dyskretne przekształcenie Fouriera (DPF).
- Wyznaczanie widma amplitudowego i fazowego na podstawie wartości uzyskanych z DPF.
- Wartości współczynników korygujących wartości widma amplitudowego do postaci pozwalającej na bezpośredni odczyt wartości amplitud poszczególnych harmonicznych z wykresu widma (dotyczy także stosowania funkcji okien).
- Uzupełnianie zerami jako technika zwiększania rozdzielczości widma.
- Nakładanie okien, jako technika zmniejszenia przecieku widma.
- Wpływ stosowania uzupełniania zerami i nakładania okien na postać widma analizowanego sygnału.
- Wyznaczanie wartości częstotliwości poszczególnych prążków widma.
- Znaczenie parametrów wywołania oraz wartości zwracanej przez funkcję `atan2` języka ANSI C.
- Pobranie adresu zmiennej w języku ANSI C.
- Implementacja kursorów na elemencie *Graph*. Zagadnienie to zostało opisane w punkcie 4.5 niniejszej instrukcji.
- Tworzenie i obsługa list wyboru na interfejsie użytkownika w środowisku LabWindows/CVI z wykorzystaniem elementu *Ring*. Zagadnienie to zostało opisane w punkcie 4.5 i 4.6 niniejszej instrukcji.

3. Program ćwiczenia

Zadanie 1

a) Wykorzystując środowisko programowania LabWindows/CVI zmodyfikuj program napisany w ramach realizacji ćwiczenia 0, aby wyznaczał zespolone widmo na podstawie wczytanych próbek sygnału. W tym celu wykorzystaj funkcję `ReFFT` znajdującą się w bibliotece Advanced Analysis wspomnianego środowiska. Jej opis znajduje się w punkcie 4.2 niniejszej instrukcji.

b) Bazując na informacjach podanych na wykładzie, zaimplementuj wyznaczanie widma amplitudowego i fazowego na bazie wartości zwracanych przez funkcję `ReFFT`. Oś odciętych wykresu widma amplitudowego i fazowego powinna być skalowana w jednostkach częstotliwości (hercach). Przy wyznaczaniu widma fazowego wykorzystaj funkcję `atan2`, a otrzymane wyniki wyraź w stopniach kątowych. Prezentację widm zrealizuj na dwóch elementach *Graph* z wykorzystaniem funkcji `PlotXY`.

c) Sprawdź poprawność napisanego programu przeprowadzając analizę widmową przebiegów zawartych w pierwszych dwóch plikach udostępnionych przez prowadzącego, gdzie znajdują się próbki zebrane w całkowitej liczbie okresów sygnału, co daje widmo bez przecieku.

d) W tworzonym programie zmodyfikuj fragment kodu realizujący wyznaczanie widma amplitudowego tak, aby wartości prążków tego widma były równe amplitudom poszczególnych harmonicznych analizowanego sygnału.

e) Dokonaj ponownego sprawdzenia poprawności działania programu porównując wyniki uzyskane dla pierwszych dwóch plików z wynikami uzyskanymi od prowadzącego zajęcia.

Zadanie 2

a) W celu umożliwienia dokładnego odczytu wartości amplitud i faz poszczególnych harmonicznych, zaimplementuj w tworzonym programie obsługę kursorów na elementach *Graph*. Implementacja tego zadania jest opisana w punkcie 4.4 niniejszej instrukcji.

Zadanie 3

a) W tworzonym programie zaimplementuj możliwość stosowania techniki uzupełnianie zerami do liczby próbek równej: 64, 128, 256, 512, 1024 oraz 2048 poprzez wykorzystanie elementu *Ring*. Podczas implementacji uwzględnij możliwość nieuzupełniania sygnału o próbki zerowe. Podczas implementacji tego zadania wykorzystaj informacje przedstawione w punkcie 4.5 i 4.6 niniejszej instrukcji.
b) Dokonaj ponownego sprawdzenia poprawności działania programu wiedząc, że w wyniku uzupełniania zerami rozdzielczość wyznaczania widma z punktu widzenia osi częstotliwości powinna zostać zwiększona. Widma uzyskane po uzupełnieniu próbek wartościami zerowymi pokaż prowadzącemu zajęcia.

Zadanie 4

a) Zmodyfikuj tworzony program, wykorzystując element *Ring*, o możliwość stosowania następujących okien czasowych: trójkątne, Hanninga, Hamminga i Blackmana. Funkcje środowiska LabWindows/CVI realizujące nakładanie okien na próbki sygnału są opisane w punkcie 4.3 niniejszej instrukcji. Podczas implementacji uwzględnij możliwość niestosowania żadnego z okien czasowych. Dla porównania przebieg sygnału przed i po nałożeniu okna powinny być prezentowane na tym samym elemencie *Graph*. Podczas implementacji tego zadania wykorzystaj informacje przedstawione w punkcie 4.5 i 4.6 niniejszej instrukcji związane z obsługą elementów *Ring*.
b) Dokonaj sprawdzenia poprawności działania programu wiedząc, że nakładanie zaimplementowanych okien powinno zmniejszyć poziom listków bocznych w obserwowanym widmie sygnału.

4. Wskazówki do ćwiczenia

4.1. Próbki sygnałów do analizy

Próbki sygnałów wykorzystywanych do badań znajdują się w plikach `WAVE2_XY.CPS`[1]. Pliki dla danej grupy ćwiczeniowej zawierają próbki tego samego sygnału okresowego jednak pobrane w całkowitej (sygnały nr 1 i 2) oraz niecałkowitej (sygnały nr 3 i 4) liczbie okresów.

4.2. Wyznaczanie dyskretnego przekształcenia Fouriera

Funkcja `ReFFT` znajdująca się w bibliotece Advanced Analysis » Signal Processing » Frequency Domain, której prototyp zamieszczono poniżej

```
ReFFT (double ArrayX_Real, double ArrayX_Imaginary, int NumberOfElements);
```

wyznacza dyskretne przekształcenie Fouriera bazując na algorytmie FFT o łączonych podstawach (ang. *split-radix algorithm*) przy założeniu, że wartości próbek analizowanego sygnału są liczbami rzeczywistymi. Poszczególne parametry funkcji mają następujące znaczenie:

`ArrayX_Real`	tablica, która przy wywołaniu funkcji powinna zawierać próbki sygnału, dla którego będzie wyznaczane widmo; na wyjściu zawiera części rzeczywiste wartości określających dyskretne widmo sygnału;
`ArrayX_Imaginary`	tablica, której wartości są ignorowane przy wywołaniu funkcji; na wyjściu zawiera części urojone wartości określających dyskretne widmo sygnału;
`NumberOfElements`	liczba próbek analizowanego sygnału, na podstawie których będzie wyznaczane jego widmo.

UWAGA: W wyniku działania funkcji `ReFFT` próbki analizowanego sygnału znajdujące się w tablica `ArrayX_Real` są nadpisywanie częściami rzeczywistymi wyznaczonego widma, dlatego aby sygnał mógł

[1] X oznacza numer grupy laboratoryjnej, natomiast Y to jednocyfrowy numer sygnału.

być poddawany wielokrotnej analizie bez ponownego wczytywania próbek z pliku, należy przewidzieć w pisanym programie działanie funkcji `ReFFT` na kopii sygnału odczytanego z pliku.

4.3. Nakładanie okien czasowych

W środowisku LabWindows/CVI (Advanced Analysis » Signal Processing » Windows) znajdują się również implementacje funkcji, które pozwalają na łatwe nakładanie okien czasowych. Są to następujące funkcje:

```
TriWin(double Samples[], int NumberOfElements);          okno trójkątne
HanWin(double Samples[], int NumberOfElements);          okno Hanninga
HamWin(double Samples[], int NumberOfElements);          okno Hamminga
BkmanWin(double Samples[], int NumberOfElements);        okno Blackmana
```

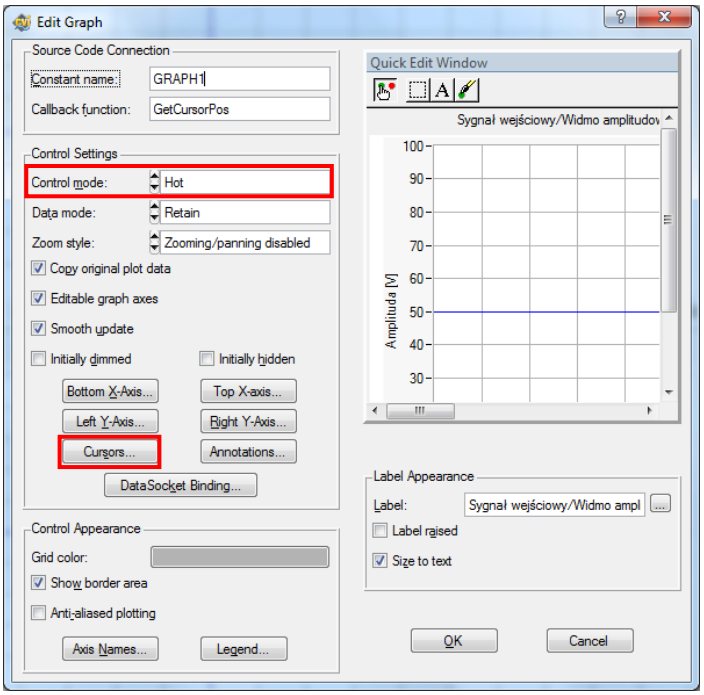
gdzie:

`Samples`	tablica próbek sygnału, na które będzie nałożone dane okno,
`NumberOfElements`	liczba próbek sygnału na które będzie nałożone okno czasowe.

UWAGA: Prezentowane funkcje modyfikują próbki analizowanego sygnału przez nałożenie odpowiedniej funkcji okna, dlatego aby sygnał mógł być poddawany wielokrotnej analizie przy wykorzystaniu różnych okien bez ponownego wczytywania próbek z pliku, należy stosować te funkcje zawsze na nową kopię sygnału odczytanego z pliku.

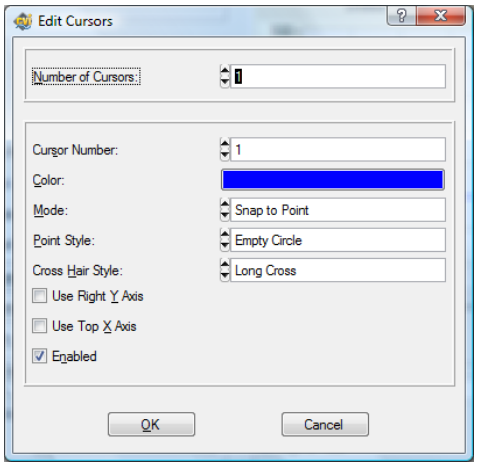
4.4. Implementacja kursorów na elemencie Graph

W środowisku LabWindows/CVI istnieje możliwość prostej implementacji kursora na elemencie *Graph* i odczytu jego położenia. Może być on wykorzystany do dokładnego odczytu wartości amplitud i faz harmonicznych analizowanego sygnału. W celu jego implementacji należy w edytorze interfejsu użytkownika wejść do ustawień elementu *Graph* i klikając klawisz Cursors… znajdujący się w grupie Control Settings (patrz rysunek 1).



Rys. 1. *Widok okna ustawień elementu Graph*

W wyniku tych działań ukarze się okno przedstawione na rysunku 2.



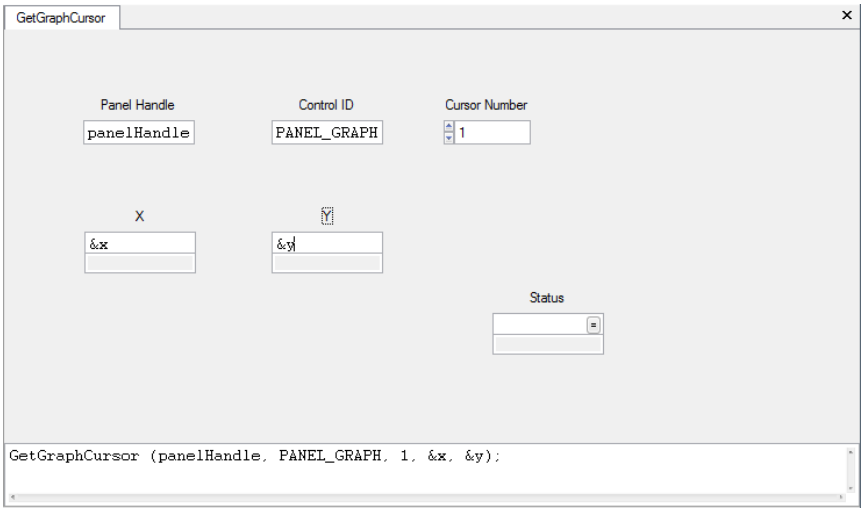
Rys. 2. *Widok okna ustawień kursorów na elemencie Graph*

W oknie tym należy dokonać ustawień jak na rysunku 2 i zaakceptować je klikając klawisz OK.

W celu umożliwienia odczytu pozycji kursora należy również w oknie ustawień elementu *Graph* w polu Callback Function wprowadzić nazwę funkcji, która zostanie wywołana, kiedy użytkownik podczas obsługi programu przemieści kursor w nowe miejsce. Dodatkowo, w ustawieniach elementu *Graph* w polu Control Mode należy wybrać opcję Hot (patrz rysunek 1), aby kliknięcie na element *Graph* uruchamiało funkcję callback. Następnie należy zamknąć okno ustawień elementu *Graph* klikając klawisz OK. Przy zaznaczonym elemencie *Graph*, którego właściwości zostały zmodyfikowane i przy kursorze umieszczonym nad nim należy wywołać menu podręczne przez kliknięcie prawego klawisza myszy i wybrać pozycję Generate Control Callback. Spowoduje to wygenerowanie szkieletu funkcji typu callback w kodzie programu.

Do prezentacji wskazania kursora należy dodatkowo na panelu tworzonej aplikacji, w edytorze interfejsu użytkownika, umieścić dwa elementy *Numeric*. Aby użytkownik nie mógł w trakcie obsługi aplikacji zmieniać wartości prezentowanych na tych elementach, należy wejść do ustawień każdego z nich i zmienić właściwość Control Mode na Indicator.

W celu implementacji obsługi wskazania kursora należy uzupełnić funkcję callback o odczyt wskazania kursora za pomocą funkcji **`GetGraphCursor`**, której okno pomocne przy uzupełnianiu parametrów przedstawiono na rysunku 3.



Rys. 3. *Okno uzupełniania parametrów funkcji* `GetGraphCursor`

Pierwsze dwa parametry należy uzupełnić zgodnie z zasadami przedstawionymi w instrukcji do ćwiczenia 1. Pole Cursor Number ustawiamy na 1, natomiast w pola X i Y wpisujemy adresy zmiennych (nazwy zmiennych poprzedzona znakiem &), do których będą zapisane współrzędne kursora przy wywołaniu tej funkcji. Następnie za pomocą funkcji **`SetCtrlVal`** wyświetlamy na umieszczonych wcześniej na panelu użytkownika elementach *Numeric* wartości wskazań kursora uzyskane w zmiennych, których adresy wprowadzono w pola X i Y wywołania funkcji **`GetGraphCursor`**.

Zakładając, że funkcję callback dla elementu *Graph* nazwano **`GetCursorPos`**, stałe wpisane w polach Constant Name identyfikujące element *Graph* oraz elementy *Numeric* to odpowiednio `GRAPH`, `NUMERIC_X`

oraz NUMERIC_Y, a zmienne, do których odczytywane są współrzędne kursora to x i y, to w efekcie opisanych działań uzyskuje się następujący kod funkcji callback elementu *Graph*.

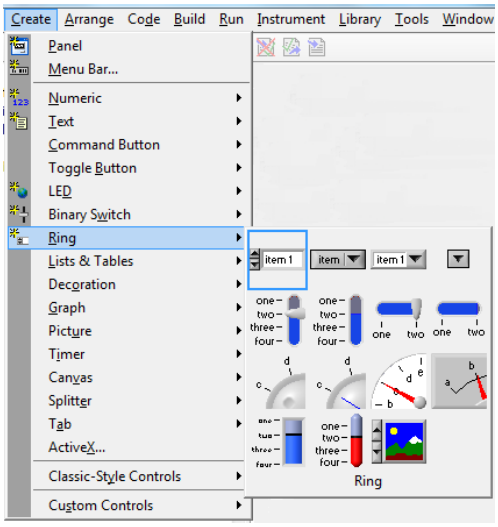
```
int CVICALLBACK GetCursorPos (int panel, int control, int event,
                              void *callbackData, int eventData1,
                              int eventData2)
{
  double x, y;

  if (event == EVENT_COMMIT)
  {
    GetGraphCursor (panelHandle, PANEL_GRAPH, 1, &x, &y);
    SetCtrlVal (panelHandle, PANEL_NUMERIC_X, x);
    SetCtrlVal (panelHandle, PANEL_NUMERIC_Y, y);
  }
  return 0;
}
```

Należy pamiętać, że implementując obsługę kursorów dla kolejnego elementu *Graph* należy przyjąć inną nazwę dla funkcji callback oraz przypisać inne stałe dla elementów *Numeric*, na których mają być wyświetlane współrzędne kurosra.

4.5. Tworzenie list wyboru na interfejsie użytkownika z wykorzystaniem elementu Ring

W środowisku LabWindows/CVI do tworzenia listy wyboru na interfejsie użytkownika można wykorzystać element typu *Ring*. Wstawienie elementu na interfejs użytkownika uzyskujemy przez wejście do menu Create i wybór pozycji Ring, a następnie z palety wybranie pierwszego elementu znajdującego się w górnym wierszu. Na rysunku 4 został on zaznaczony niebieską ramką.

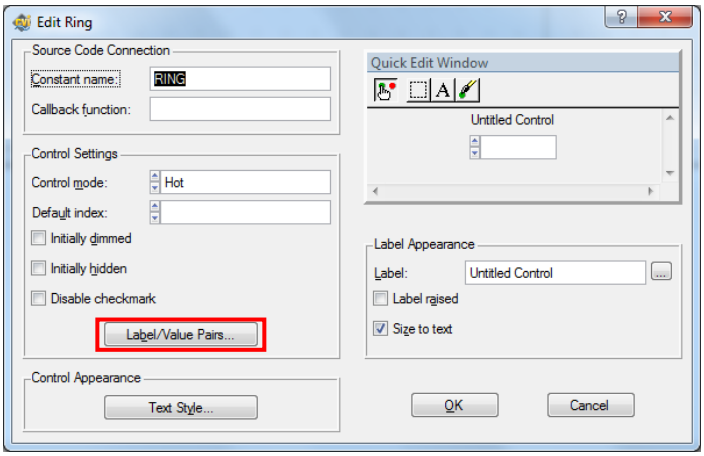


Rys. 4. *Widok palety kontrolek typu Ring*

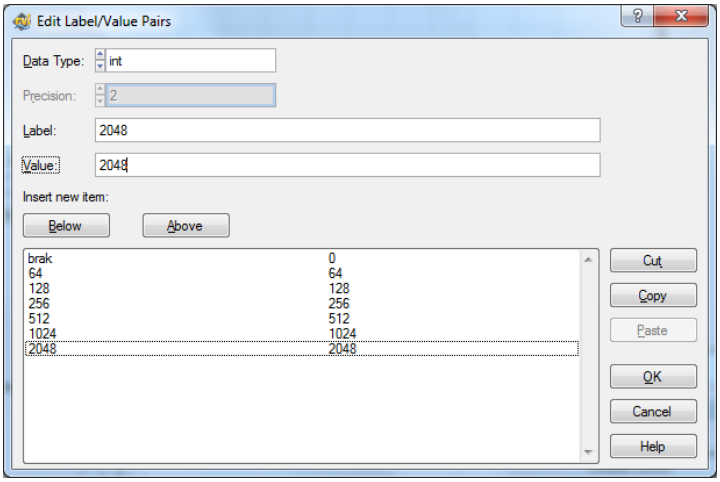
Klikając dwukrotnie na element *Ring* ułożony już na panelu użytkownika tworzonej aplikacji powodujemy otwarcie okna zmiany jego ustawień, które pokazano na rysunku 5. W celu zdefiniowania pozycji listy należy kliknąć na klawisz Label/Value Pairs. Na rysunku 5 został on wyróżniony czerwoną ramką. Po kliknięciu wskazanego klawisza zostanie otwarte okno, w którym widnieją m.in. dwa pola: Label i Value. Umożliwiają one edycję etykiety (ang. *label*) danej pozycji na liście, która będzie widoczna dla użytkownika oraz przyporządkowanej jej wartości (ang. *value*), która będzie zwracana przez element *Ring*, jeśli dana pozycja zostanie wybrana przez użytkownika. Po zdefiniowaniu pozycji listy, kolejną pozycję można dodać poniżej (klikając klawisz Below) lub powyżej (klikając klawisz Above) pozycji zaznaczonej na liście znajdującej się poniżej wymienionych klawiszy.

Edycja pozycji listy w sensie usunięcia danej pozycji, jej skopiowania lub wstawienia odbywa się odpowiednio za pomocą klawiszy Cut, Copy i Paste znajdujących na prawo od listy aktualnie zdefiniowanych pozycji listy. Zatwierdzenie zmiana związanych z edycją całej listy odbywa się przez wciśnięcie klawisza OK.

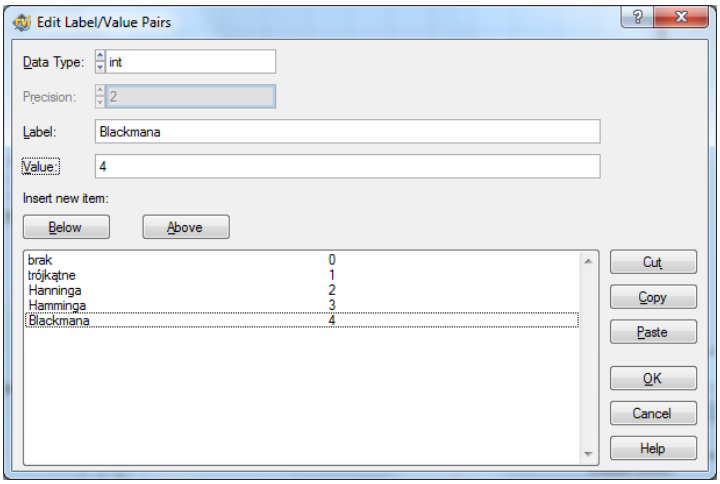
Na rysunkach 6 i 7 znajdują się przykłady zdefiniowanych list wymagane przy implementacji uzupełnienia zerami oraz nakładania okien czasowych.



Rys. 5. *Widok okna zmiany ustawień elementu typu Ring*

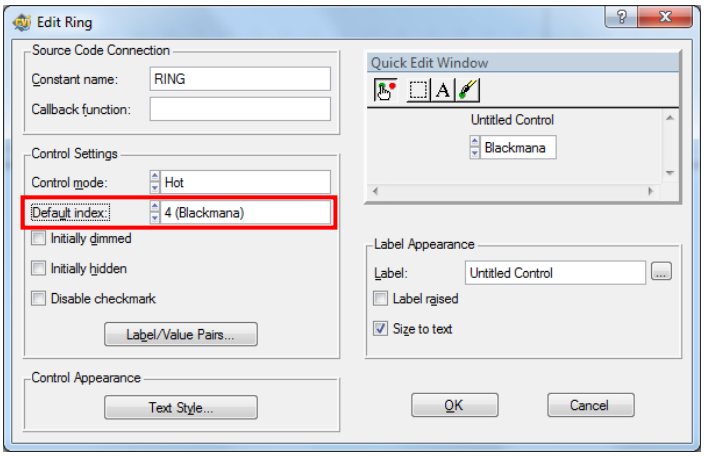


Rys. 6. *Widok okna definiowania pozycji listy w elemencie Ring dla uzupełnienia zerami*

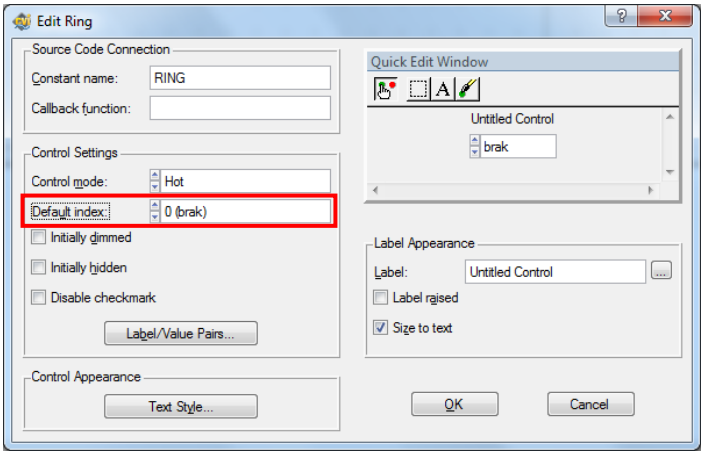


Rys. 7. *Widok okna definiowania pozycji listy w elemencie Ring dla wyboru okna czasowego*

Po zdefiniowaniu listy w elemencie *Ring*, domyślnie wybrana będzie ostatnia wprowadzona do listy pozycja. Aby to zmienić, należy z rozwijanej listy opisanej Default index wybrać inną pozycję, która ma się stać pozycją domyślną. Zmianę tą pokazują rysunki 8 i 9.



Rys. 8. *Widok okna zmiany ustawień elementu typu Ring z zaznaczoną jako domyślny wybór pozycją 4 z listy*



Rys. 9. *Widok okna zmiany ustawień elementu typu Ring z zaznaczoną jako domyślny wybór pozycją 0 z listy*

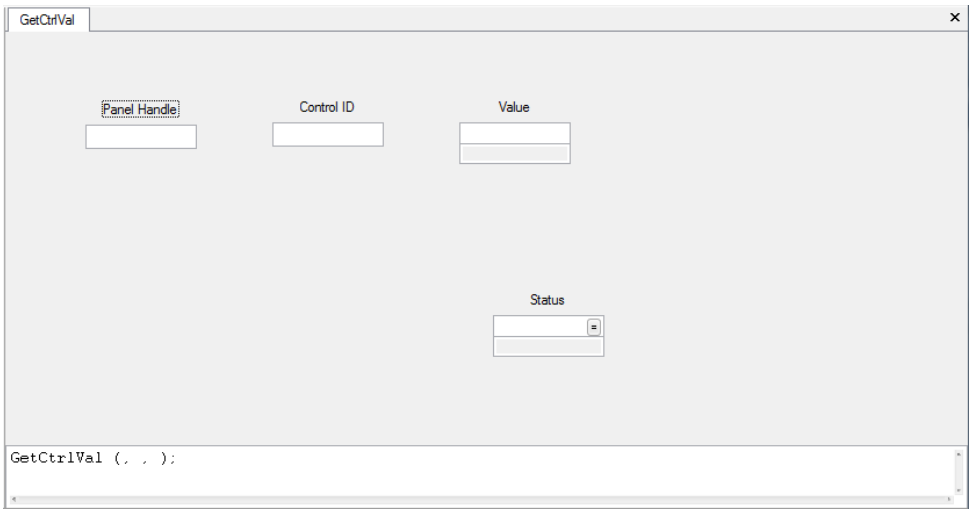
Po wprowadzeniu wszystkich zmian należy zatwierdzić je klikając klawisz OK.

4.6. Odczyt informacji z listy wyboru realizowanej z wykorzystaniem elementu Ring

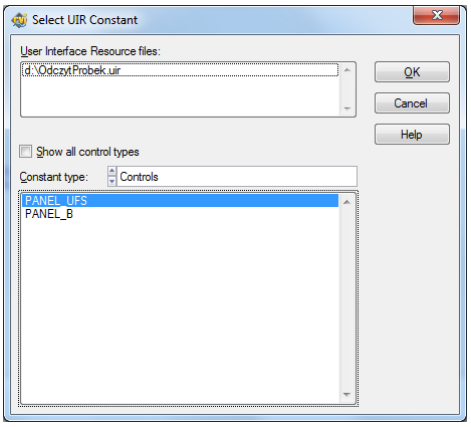
Odczyt informacji z elementu *Ring* odnośnie aktualnie wybranej pozycji listy odbywa się z wykorzystaniem funkcji GetCtrlVal. Okno pomocne przy uzupełnianiu parametrów wywołania tej funkcji pokazano na rysunku 10. Jego wywołanie uzyskuje się przez zaznaczenie nazwy funkcji w edytorze kodu programu środowiska LabWindows/CVI i wciśnięcie kombinacji klawiszy Ctrl+P.

W polu Panel Handle wprowadzamy nazwę zmiennej, która przechowuje uchwyt do panelu, na którym znajduje się element *Ring*, do którego będziemy się odwoływali. W większości przypadków programy tworzone na niniejszym laboratorium będą składały się tylko z jednego panelu (okna aplikacji). Zatem jeśli nie dokonywano żadnych zmian w polu Panel Variable Name podczas generacji szkieletu kodu aplikacji, to zmienna ta nosi nazwę `panelHandle`.

W polu Control ID należy wpisać identyfikator elementu, do którego ma odnosić się dana funkcja. Załóżmy, że w naszym przypadku jednemu z elementów *Ring* został przypisany identyfikator o nazwie `OKNO`, jednak w rzeczywistości pełna nazwa identyfikatora elementu tworzona jest z identyfikatora przypisanego panelowi (domyślnie `PANEL`) oraz identyfikatora przyporządkowanego samemu elementowi. Obydwie te nazwy są rozdzielane znakiem podkreślnika. Zatem w tym przypadku identyfikatorem elementu *Ring* będzie ciąg znakowy `PANEL_OKNO`. W przypadku, gdy nie jesteśmy pewni co do przypisanej elementowi nazwy identyfikatora, należy ustawić kursor w polu Control ID i wcisnąć klawisz  znajdujący się w pasku narzędzi okna (lub klawisz Enter), co powoduje otwarcie okna pokazanego na rysunku 11. Pozwala ono na wybór i wprowadzenie właściwej, a co najważniejsze poprawnej, nazwy identyfikującej element.



Rys. 10. *Okno uzupełniania parametrów wywołania funkcji* `GetCtrlVal`



Rys. 11. *Okno z nazwami identyfikatorów elementów użytych w tworzonej aplikacji*

Na koniec w polu Value należy wprowadzić adres zmiennej, do której zostanie zapisana wartość odczytana z elementu *Ring*. Wartością tą będzie wartość przypisana aktualnie wybranej pozycji podczas definiowania wszystkich pozycji listy (patrz rysunek 6 lub 7).

Wprowadzenie uzupełnionego wywołania funkcji do kodu aplikacji dokonujemy klawiszem  .

5. Opracowanie wyników

Sprawozdanie z ćwiczenia powinno zawierać wymienione poniżej elementy ***wraz z komentarzami słownymi***.

- Widok interfejsu użytkownika stworzonej aplikacji i krótki opis jej obsługi.
- Zebrane w tabeli numery oraz wartości częstotliwości, amplitud i faz poszczególnych harmonicznych analizowanego sygnału na podstawie analizy próbek znajdujących się w pliku 1 lub 2.
- Wnioski odnośnie wpływu całkowitej i niecałkowitej liczby próbkowanych okresów sygnału na poprawność interpretacji wyznaczonego w oparciu o te próbki widma amplitudowego. Wnioski należy poprzeć przykładem widm wyznaczonych na podstawie próbek zebranych w całkowitej (plik 1 lub 2) i niecałkowitej (plik 3 lub 4) liczbie jego okresów.
- Dla przypadku zebrania próbek w niecałkowitej liczbie okresów sygnału (plik 4) omówić wpływ stosowania uzupełniania zerami na wygląd widma amplitudowego i dokładność wyznaczania harmonicznych sygnału. Omówienia należy poprzeć przykładami, gdzie nie zastosowano uzupełniania zerami oraz dwoma przykładami z zastosowaniem uzupełniania zerami.
- Dla przypadku zebrania próbek w niecałkowitej liczbie okresów sygnału (plik 4) i zastosowania uzupełniania zerami do dużej liczby próbek przedstawić i omówić wpływ zastosowania różnych okien czasowych na tłumienie lisków bocznych w widmie amplitudowym sygnału.
- W przypadku plików 3 i 4 określić liczbę całkowitych okresów sygnału znajdujących się w zakresie zebranych próbek. Następnie, bazując na próbkach znajdujących się w wymienionych powyżej plikach, należy wyznaczyć widma amplitudowe, jednocześnie stosując okno Blackmanna oraz uzupełnianie

zerami do dużej liczby próbek. Na podstawie uzyskanych widm wyciągnąć wnioski odnośnie wpływu liczby próbkowanych okresów, przy jednoczesnym stosowaniu technik uzupełniania zerami i nakładania okien, na poprawność interpretacji widma amplitudowego i dokładność wyznaczania składowych harmonicznych przy akwizycji niecałkowitej liczby okresów analizowanego sygnału.

Na końcową ocenę z ćwiczenia mają również wpływ:

- systematyczna realizacja poszczególnych zadań ćwiczenia;
- poprawne skomentowanie kodu całej tworzonej aplikacji;
- poprawny styl kodowania (stosowanie wcięć w kodzie itp.);
- bezbłędne działanie stworzonego programu przy różnych niestandardowych działaniach użytkownika;
- implementacja wszystkich założeń dla aplikacji przedstawionych przez prowadzącego zajęcia;
- końcowy wygląd interfejsu użytkownika aplikacji, który powinien być estetyczny i przejrzysty;
- przejrzyste i estetyczne formatowanie pisemnego sprawozdania z ćwiczenia.

6. Literatura

[1] Borodziewicz W., Jaszczak K.: „*Cyfrowe przetwarzanie sygnałów – wybrane zagadnienia*". WNT, Warszawa 1987.

[2] Dąbrowski A.: „*Przetwarzanie sygnałów przy użyciu procesorów sygnałowych*". Wydawnictwo Politechniki Poznańskiej 2000.

[3] Lyons R.G.: „*Wprowadzenie do cyfrowego przetwarzania sygnałów*". WKŁ, Warszawa 1999.

[4] Oppenheim V, Schaffer R.W.: „*Cyfrowe przetwarzanie sygnałów*". WKŁ, Warszawa 1979.

[5] Oppenheim V, Willsky A.S.: „*Signals & Systems*". Prentice Hall, 1997.

[6] Oppenheim V, Schaffer R.W.: „*Discrete-Time Signal Processing*". WKŁ, Warszawa 1979.

[7] Roberts R.A., Mullis C.T.: „*Digital Signal Processing*". Addison-Wesley Publishing Company, 1987.

[8] Stabrowski M.: „*Miernictwo elektryczne. Cyfrowa technika pomiarowa*". Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 1994.

[9] Sydenham P.H.: „*Podręcznik metrologii – tom I*". WKŁ, Warszawa 1988.

[10] Zieliński T.P.: „*Od teorii do cyfrowego przetwarzania sygnałów*". Wydział EAIiE AGH Kraków 2002.

[11] Zieliński T.P.: „*Cyfrowe przetwarzanie sygnałów. Od teorii do zastosowań*". WKŁ, Warszawa 2005.