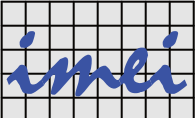
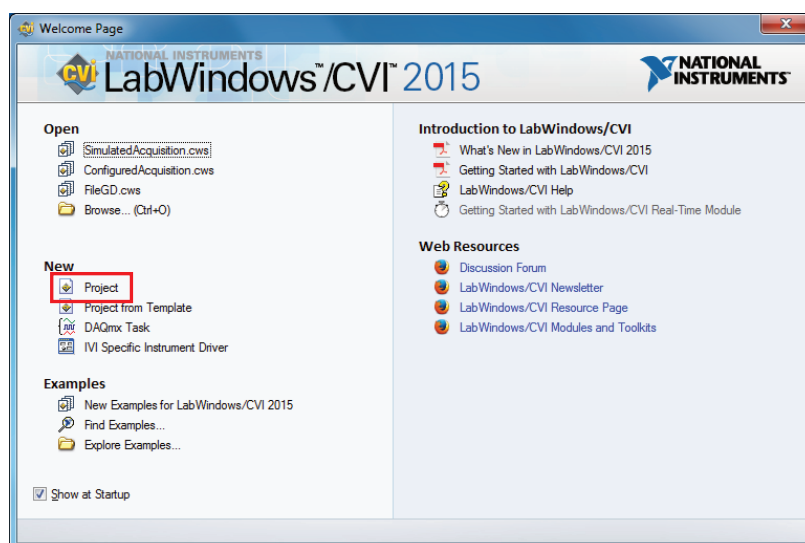


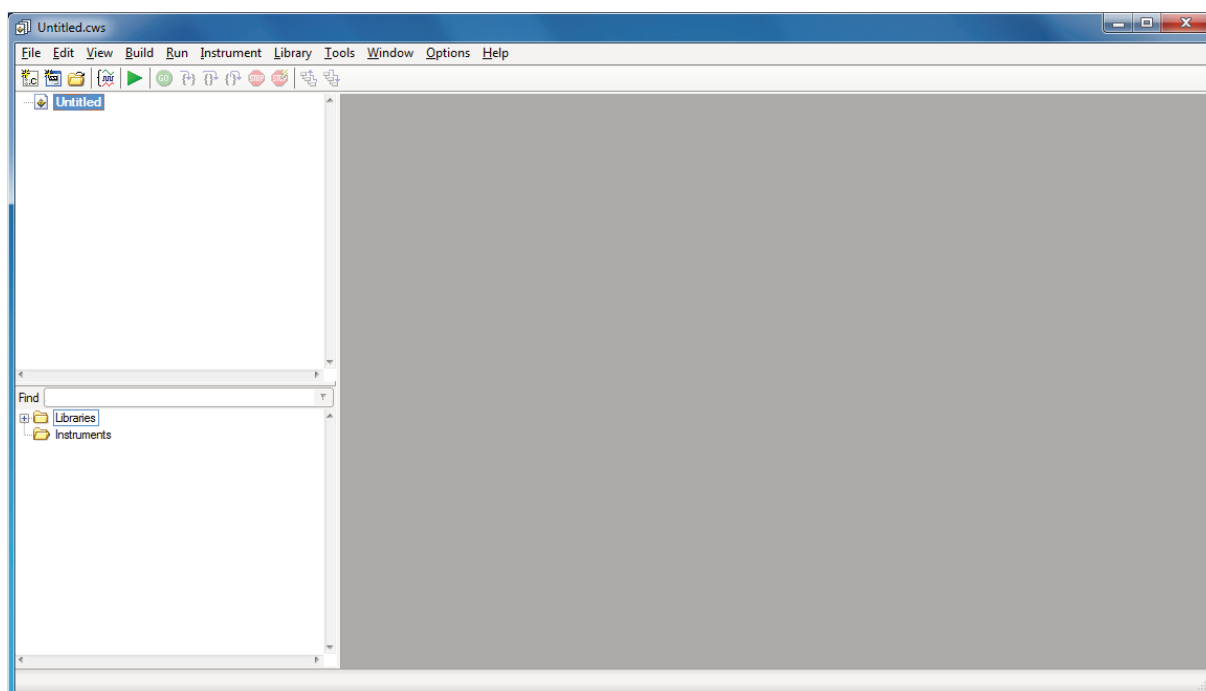
| | |
|---|--|
| CYFROWE PRZETWARZANIE SYGNAŁÓW Laboratorium |  Instytut Metrologii, Elektroniki i Informatyki |
| Temat: Tworzenie aplikacji w środowisku LabWindows/CVI | Wprowadzenie |

Tworzenie nowej aplikacji w środowisku LabWindows/CVI rozpoczynamy od otwarcia nowego projektu. W tym celu, w oknie przedstawionym na rysunku 1 klikamy na pozycję Project w grupie New.



Rys. 1. Wygląd głównego okna środowiska LabWindows/CVI po jego uruchomieniu

Wymienione powyżej działanie prowadzi do uzyskania okna środowiska jak na rysunku 2.


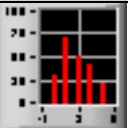


Rys. 2. Wygląd okna środowiska LabWindows/CVI po otwarciu nowego projektu

Przy tworzeniu nowej aplikacji, w pierwszej kolejności należy przystąpić do tworzenia interfejsu użytkownika. W tym celu z menu File Wybieramy New | User Interface (*.uir)... Rozmiary uzyskanego w ten sposób szarego panelu z napisem „Untitled Panel”, który jest podstawą interfejsu użytkownika tworzonej aplikacji, można dowolnie zmieniać. W tym celu należy umieścić kursor nad jedną z jego krawędzi i przytrzymując lewy klawisz myszy dokonać pożądanych zmian.

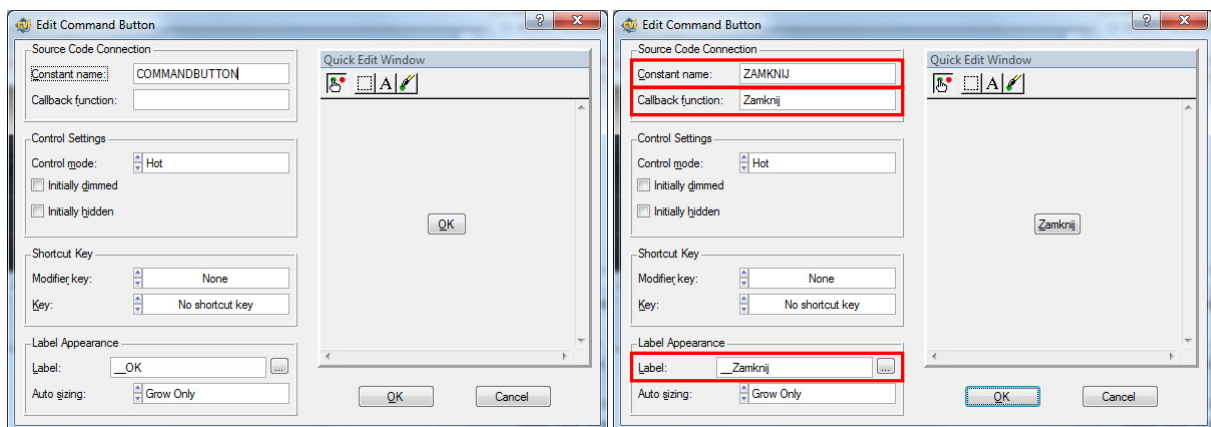
W następnej kolejności należy umieścić na panelu pożądane elementy. W niniejszym programie będą to dwa elementy typu *Square Command Button* oraz element *Graph*. Miejsce ich położenia w menu środowiska LabWindows/CVI przedstawia tabela 1.

Tabela 1. Położenie elementów wymaganych na interfejsie użytkownika tworzonej aplikacji

| Element | Ikona elementu | Położenie w menu |
|-----------------------|---|---|
| Square Command Button |  | Create Command Button Square Command Button |
| Graph |  | Create Graph Graph |

Po ułożeniu elementów na panelu można dokonać zmiany ich rozmiarów. Dotyczy to szczególnie elementu *Graph*, który będzie odpowiadał za graficzną prezentację sygnału, dlatego powinien być on na tyle duży, aby wyraźnie go prezentował. W tym celu należy kliknąć na dany element, umieścić kursor nad jednym z czarnych kwadracików, które pojawiły się na jego rogach i trzymając wciśnięty lewy klawisz myszy dokonać odpowiednich zmian.

Sygnał na elemencie *Graph* będzie mógł być zaprezentowany tylko w wyniku wcześniejszego wczytania jego próbek z pliku. Programowa realizacja tego zadania będzie inicjowana przez naciśnięcie jednego z dwóch klawiszy *Square Command Button*. Drugi klawisz będzie odpowiadał za zamknięcie aplikacji. Dlatego teraz konieczna jest zmiana niektórych właściwości tych elementów. W tym celu klikamy dwukrotnie na jeden z elementów *Square Command Button*, powodując pojawienie się okna z jego domyślnymi ustawieniami pokazanymi po lewej stronie rysunku 3.



Rys. 3. Wygląd okna ustawień parametrów elementu *Square Command Button* przed (po lewej) i po (po prawej) dokonaniu niezbędnych zmian

W ustawieniach tego elementu zostaną dokonane tylko trzy niezbędne zmiany. Pierwsza dotyczy parametru Constant Name, który jednoznacznie identyfikuje element na panelu, umożliwiając później programowe odwołanie się do niego z kodu aplikacji. Constant Name może być dowolnym ciągiem znakowym (bez stosowania polskich liter). Przy jego wprowadzaniu należy używać tylko dużych liter alfabetu, cyfr i znaku podkreślenia. Ciąg ten jako nazwa jednocześnie powinien opisywać przeznaczenie elementu. Zakładając, że klawisz ten będzie służył do zamykania programu, dlatego w niniejszym przypadku w omawianym polu wprowadzono nazwę ZAMKNIJ.

Drugim parametrem podlegającym zmianie jest pole Callback Function, gdzie wprowadzamy nazwę tzw. funkcji callbackowej, która będzie związana z danym elementem i wywoływana w przypadku realizacji przez użytkownika aplikacji jakichkolwiek działań na tym elemencie, np. kliknięcie. Tworzenie nazw funkcji

callbackowych podlega zasadom tworzenia nazw funkcji w języku C, dlatego w naszym przypadku może być to np. nazwa Zamknij.

Trzecim parametrem jest Label, czyli etykieta, która pojawia się na edytowanym przycisku wskazując późniejszemu użytkownikowi aplikacji jego funkcjonalne przeznaczenie. W tym miejscu można wprowadzić dowolne wyrazy oddzielone spacjami, w których można stosować polskie znaki. W naszym przypadku może to być np. Zamknij.

Po prawej stronie rysunku 3 pokazano panel właściwości elementu *Square Command Button* z wprowadzonymi ustawieniami. Akceptacja wprowadzonych zmian odbywa się przez wciśnięcie przycisku OK.

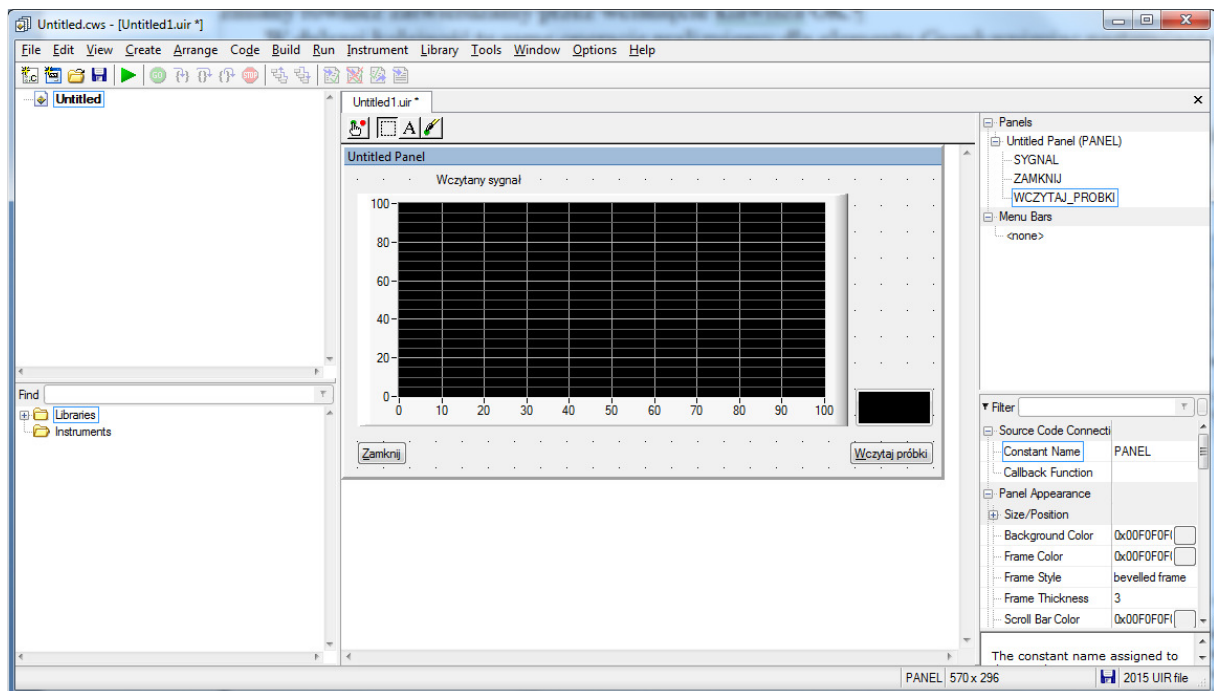
Identycznie klikając na drugi z przycisków wchodzimy do ustawień jego parametrów. Podobnie zmieniamy Constant Name tym razem np. na W CZYT AJ _ P RO B K I, a w polu Label wpisujemy np. Wczytaj próbki, ponieważ klawisz ten będzie służył inicjalizacji procesu wczytywania próbek sygnału z pliku. Dodatkowo w polu Callback Function wprowadzamy nazwę funkcji callbackowej, np. WczytajProbki. Wprowadzone zmiany również zatwierdzamy przez wciśnięcie klawisza OK.

W dalszej kolejności te same operacje realizujemy dla elementu *Graph* wpisując następujące parametry:

- pole Constant Name: SYGNAL,
- pole Label: Wczytany sygnał.

Tym razem pole związane z funkcją callbackową pozostawiamy puste, gdyż z punktu widzenia aplikacji nie chcemy reagować na zdarzenia generowane przez użytkownika na tym elemencie.

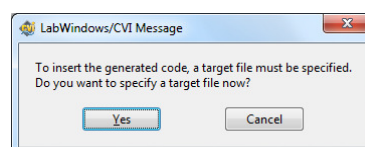
Uzyskujemy w ten sposób panel interfejsu użytkownika tworzonej aplikacji przedstawiony na rysunku 4.



Rys. 4. Wygląd interfejsu użytkownika tworzonej aplikacji

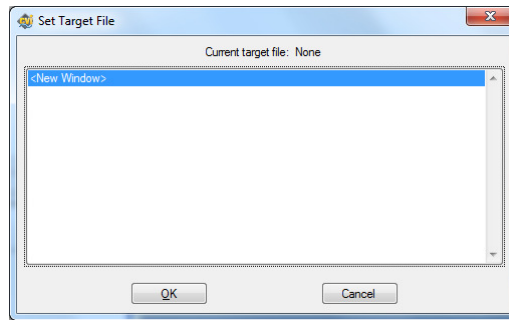
Tak stworzony panel zapisujemy wybierając z menu File | Save Untitled1.uir As... Przy zapisie pliku należy zapamiętać podaną nazwę, gdyż będzie ona potrzebna w dalszej części tworzenia programu.

Kolejnym krokiem jest generacja szkieletu kodu aplikacji. W tym celu z menu Code wybieramy Generate | All Code..., W niektórych przypadkach powoduje to pojawienie się okna z komunikatem, jak na rysunku 5, gdzie użytkownik jest pytany, czy chce podać plik docelowy, w którym będzie wygenerowany szkielet kodu aplikacji. Na tym etapie tworzenia aplikacji należy wcisnąć klawisz Yes.



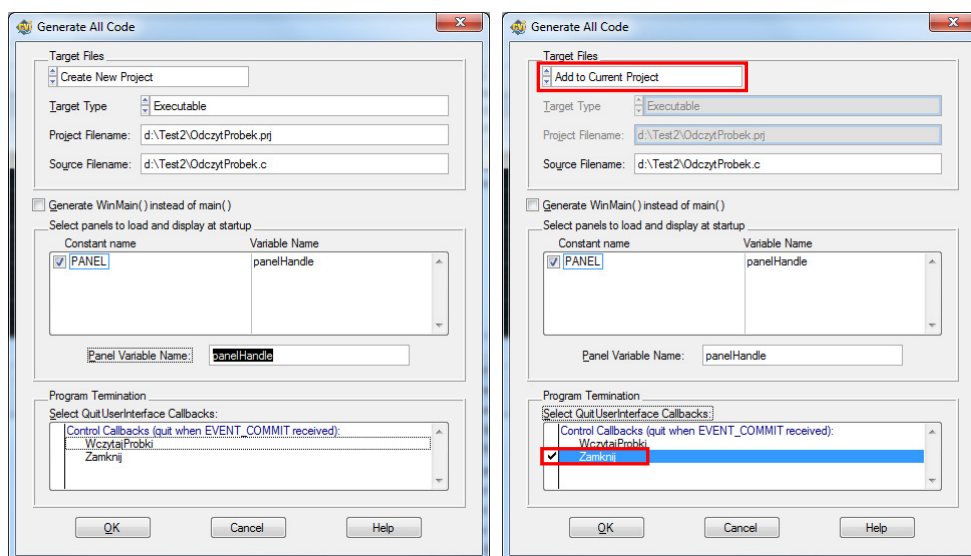
Rys. 5. Komunikat przy generowaniu szkieletu kodu aplikacji

Wciśnięcie klawisza Yes powoduje pojawienie się okna przedstawionego na rysunku 6, w którym właściwie powinniśmy wcisnąć klawisz OK, akceptując tym samym generowanie szkieletu kodu aplikacji w nowym oknie.



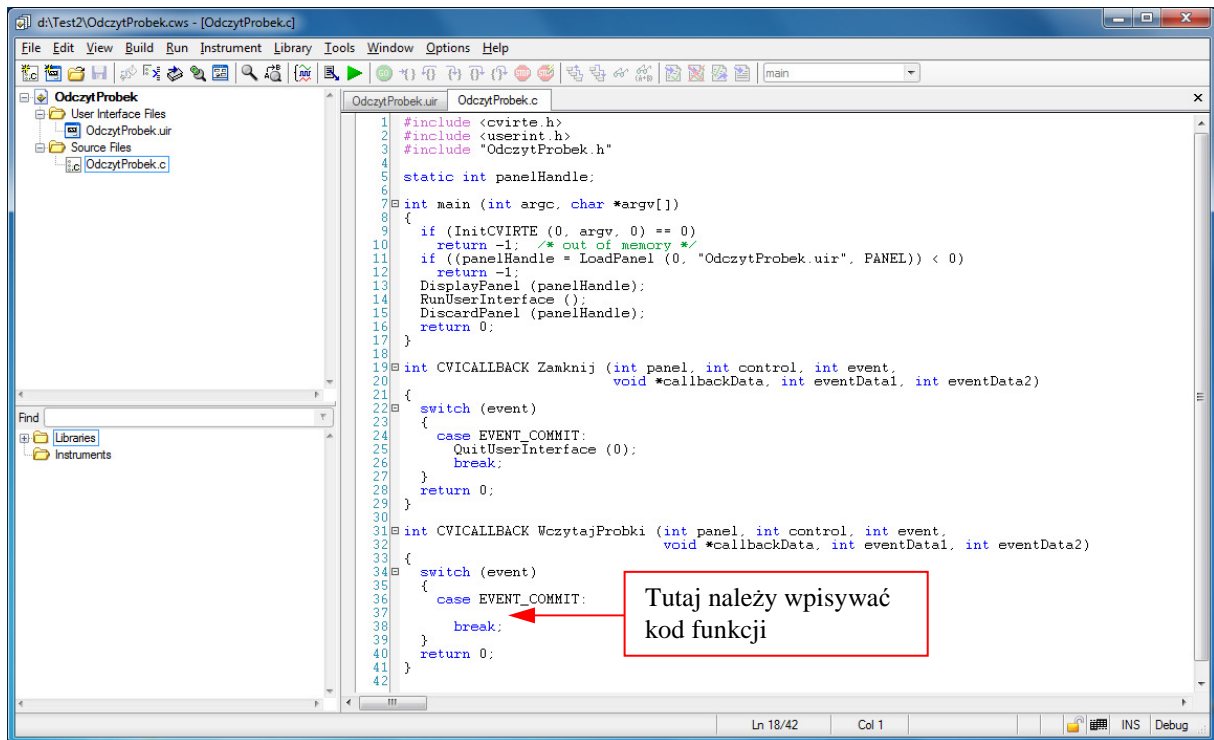
Rys. 6. Okno wyboru pliku, w którym będzie wstawiony szkielet kodu tworzonej aplikacji

Opisane działanie spowoduje pojawienie się okna przedstawionego po lewej stronie rysunku 7.



Rys. 7 Okno generacji szkieletu kodu aplikacji przed i po wprowadzaniu niezbędnych zmian

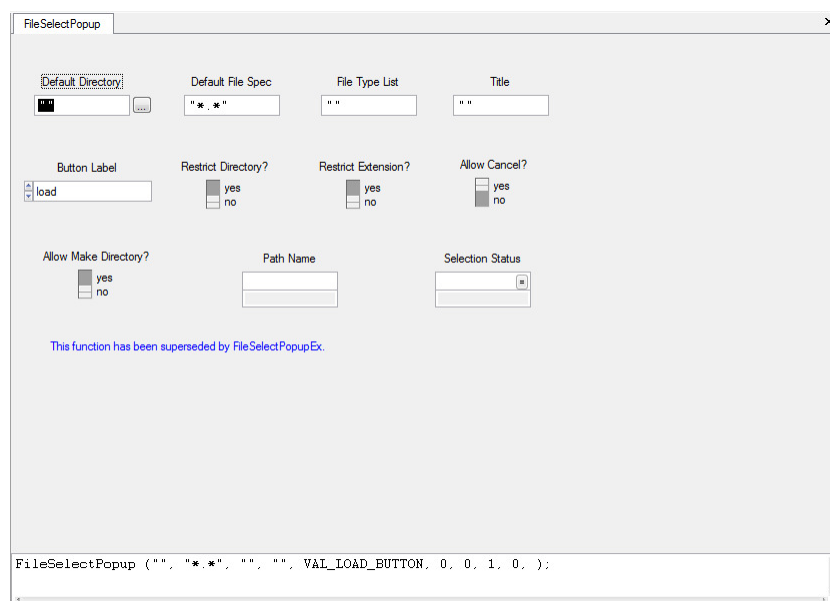
W polu, w którym widnieje napis Create New Project wybieramy Add to Current Project, ponieważ na początku nowy projekt został już stworzony, więc nie jest konieczne robić to po raz drugi. W polu Select QuitUserInterface Callbacks należy zaznaczyć nazwę funkcji callbackowej, która będzie realizowała operację wyjścia z programu. W naszym przypadku jest to funkcja Zamknij (przypisana klawiszowi z napisem Zamknij). Wspomniane zaznaczenie odbywa się na wąskim pasku po lewej stronie omawianego pola (patrz okno po prawej stronie na rysunku 7). Wprowadzone zmiany należy zaakceptować wciskając klawisz OK. Powoduje to wygenerowanie szkieletu kodu aplikacji (patrz rysunek 8) oraz automatyczne otwarcie okna w celu zapisania całego stworzonego projektu. Projekt należy zapisać pod taką samą nazwą, jak plik z interfejsem użytkownika. Umożliwi to później łatwe odnalezienie wszystkich plików danej aplikacji.




Rys. 8. Wygenerowany szkielet kodu aplikacji

Jak widać, funkcja `Zamknij` nie wymaga właściwie żadnych modyfikacji. W jej ciele jest wywoływana funkcja biblioteczna `QuitUserInterface`, która zamyka aplikację. Natomiast funkcja `Wczytaj` wymaga dopisania fragmentu kodu, który umożliwi wczytanie próbek sygnałów zapisanych w plikach `*.cps`. Kod ten musi zostać wprowadzony pomiędzy linie `case EVENT_COMMIT: a break;`, tj. począwszy od 37 linii kodu.

Pierwszą rzeczą przy uzupełnianiu kodu funkcji `WczytajProbki` jest programowe wywołanie standardowego okna wyboru pliku w systemie Windows, które umożliwi w sposób graficzny określenie ścieżki dostępu do pożądanego pliku. W tym celu we wspomnianym miejscu rozpoczęcia implementacji ciała funkcji wpisujemy nazwę funkcji wywołującej takie okno, tj. `FileSelectPopup`. Następnie zaznaczamy wpisaną nazwę i wciskamy kombinację klawiszy `Ctrl+P`, która otwiera okno pomagające uzupełnić parametry wywołania tej funkcji (rysunek 9).



Rys.9. Okno uzupełniania parametrów funkcji `FileSelectPopup`

W otwartym oknie zmieniamy tylko wybrane parametry. Jednym z nich jest Default File Spec, który określa, z jakimi rozszerzeniami pliki będą widoczne po otwarciu okna. W polu tym wpisujemy "*.cps". W polu Title wpisujemy nazwę, jaka ma pojawić się na niebieskim pasku u góry otwartego okna, wskazując użytkownikowi aplikacji czynność do wykonania. W naszym przypadku może to być "Wybierz plik z próbkami". W polu Path Name wpisujemy nazwę zmiennej, do której zostanie zapisana ścieżka dostępu do wybranego pliku (może być to np. nazwa ścieżka). Natomiast w polu Selection Status wpisujemy nazwę zmiennej, do której zostanie zapisana wartość zwracana przez funkcję FileSelectPopup (może być to np. nazwa status). Zarówno zmienna ścieżka jak i status zostaną zadeklarowane jako typowe zmienne w języku C, dlatego nie należy używać w ich nazwach polskich znaków. W celu wprowadzenia uzupełnionego wywołania niniejszej funkcji, które widać u dołu okna, należy wcisnąć klawisz  znajdujący się na pasku narzędzi umieszczonym w górnej części okna, a w kolejnym otwartym oknie klawisz Replace. W tym momencie można już zamknąć okno uzupełniania wywołania funkcji lub też zostanie ona automatycznie zamknięta.

Wspomniane w powyższym akapicie dwie zmienne należy zadeklarować jako zmienne lokalne funkcji WczytajProbki, przyporządkowując im typy podane poniżej.

```
int status;
char sciezka[MAX_PATHNAME_LEN];
```

MAX_PATHNAME_LEN jest stałą predefiniowaną w środowisku LabWindows/CVI, która określa maksymalną możliwą długość ścieżki do pliku w systemie Windows. Dzięki jej użyciu możemy zadeklarować tablicę, która na pewno pomieści dowolną możliwą ścieżkę do pliku dopuszczalną tym systemie.

Programowa realizacja dalszych operacji związanych z odczytem próbek z pliku ma sens tylko wtedy, gdy funkcja FileSelectPopup zwróci wartość równą 1, gdyż zgodnie z dokumentacją funkcji oznacza to, iż został wybrany jakiś istniejący plik. W przeciwnym wypadku albo nie wybrano żadnego pliku, albo wcisnięto klawisz Anuluj. Zatem kolejne linie kodu po wywołaniu funkcji FileSelectPopup będą miały postać jak poniżej.

```
if (status == 1)
{
}
}
```

Kolejnymi operacjami, jakie należy wykonać w ramach powyższej instrukcji if jest: otwarcie pliku do odczytu, odczytanie z niego danych i zamknięcie pliku. Zadania te studenci powinni wykonać samodzielnie w ramach ćwiczenia 0. Wszystkie dane zawarte w pliku należy zapamiętać w zmiennych odpowiedniego typu, aby mieć możliwość zarówno ich późniejszego wykorzystania jak również prezentacji na panelu użytkownika.

Postać funkcji WczytajProbki po wszystkich modyfikacjach zaprezentowanych w niniejszym dokumencie powinna wyglądać jak poniżej. Wypunktowanymi komentarzami przedstawiono zadania, jakie należy wykonać w ramach ćwiczenia 0.

```
int CVICALLBACK WczytajProbki (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    int status;
    char sciezka[MAX_PATHNAME_LEN];


    switch (event)
    {
        case EVENT_COMMIT:
            // Określenie ścieżki dostępu do pliku
            status = FileSelectPopup ("", "*.cps", "", "Wybierz plik z próbkami",
                VAL_LOAD_BUTTON, 0, 0, 1, 0, sciezka);

            if (status == 1)
            {
                // 1. Otwarcie pliku binarnego do odczytu
                // 2. Odczyt początkowych danych z pliku
                // 3. Wyświetlenie odczytanych parametrów
                // 4. Przydzielenie pamięci dla tablic
                // 5. Odczyt kodów próbek
                // 6. Zamknięcie pliku
            }
        }
    }
}
```

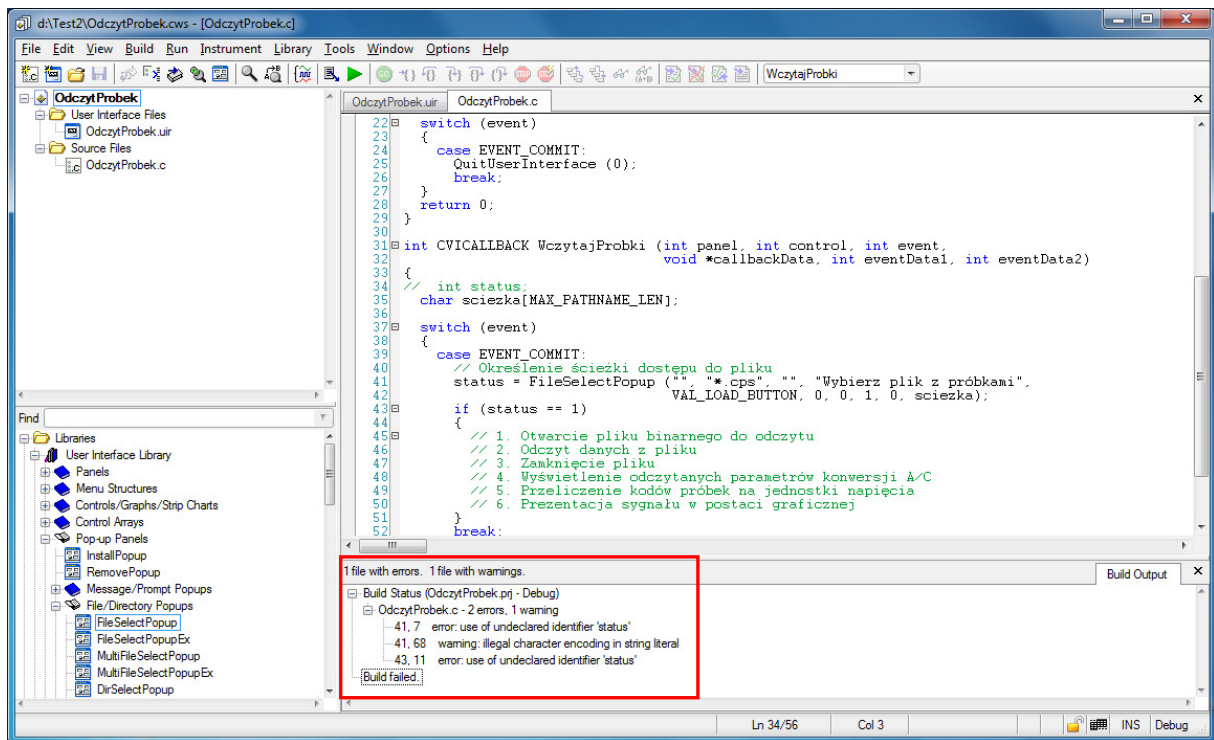
```

// 7. Przeliczenie kodów próbek na jednostki napięcia
// 8. Prezentacja sygnału w postaci graficznej
// 9. Zwolnienie pamięci zajmowanej przez tablice
}
break;
}
return 0;
}
}

```

W celu sprawdzenia poprawności wykonania opisanych działań, należy uruchomić aplikację. W tym celu klikamy ikonę , co powinno spowodować uruchomienie stworzonej aplikacji przez pojawienie się na monitorze jej interfejsu użytkownika. W przypadku kliknięcia na klawisz opisany Wczytaj próbki powinno nastąpić otwarcie standardowego okna środowiska Windows umożliwiającego wskazanie pliku.

W przypadku wystąpienia błędów kompilacji program nie zostanie uruchomiony. Na rysunku 10 przedstawiono widok środowiska LabWindows/CVI w przypadku kompilacji programu zakończonej niepowodzeniem. W oknie znajdującym się w dolnej części środowiska znajduje się informacja o niepowodzeniu kompilacji i jej przyczynach. W takim przypadku należy przeczytać uważnie informacje opatrzone słowem error i warning i usunąć zaszygalizowane błędy.



Rys.10. Widok okna środowiska LabWindows/CVI w przypadku braku poprawnej kompilacji programu