Podstawy programowania w LabView (4)

Operacje na zmiennych znakowych, pliki tekstowe, zasady tworzenia wykresów

Do tej pory poznaliśmy środowisko programistyczne, podstawowe elementy języka programowania graficznego, pracę z tablicami i złożony typ jakim jest Klaster Danych. W tej części zajmiemy się pracą ze zmiennymi znakowymi, obsługą plików tekstowych oraz zasadami tworzenia wykresów.

Operacje na zmiennych znakowych należą do jednych z najczęściej wykonywanych. Gdyż jest to czytelny dla ludzi sposób przekazania informacji. W postaci wymiany komunikatów tekstowych często przebiega komunikacja pomiędzy komputerem i podłączonym do niego urządzeniem pomiarowym. Komunikaty te zwykle zgodne są z językiem SCPI. Dlatego operacjom na zmiennych znakowych należy poświęcić trochę uwagi. W LabView zmienne znakowe określane są jako typ String oznaczający łańcuch znaków. Do jego obsługi przygotowano paletę String.

Paleta funkcji String

Wszystkie niezbędne funkcje zostały zgromadzane w palecie *String* pokazanej na **rysunku 38**. Oprócz podstawowego zestawu funkcji znajdziemy tam trzy dodatkowe palety:

- *Additional String Functions* zawierającą dodatkowe funkcje operujące na tekście.
- *String Number Conversion* zawierającą funkcje konwertujące dane tekstowe do numerycznych.
- *String/Array/Path Conversion* zawierającą funkcje do konwersji typu *string* na *Path*, czyli ścieżkę dostępu do pliku, tablicę i odwrotnie.

String Length – zwraca liczbę znaków w tekście.

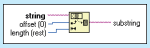
string **– badany tekst, może to być tablica tekstowa, klaster danych tekstowych, lub tablica klastrów danych tekstowych.**

length **– liczba znaków w tekście, struktura zmiennej jest taka sama jak wejściowej string.**

Concatenate Strings - łączy teksty wejściowe i jednowymiarowe tablice tekstowe w jeden tekst, zmiany liczby wejść można dokonać przez rozciągnięcie funkcji lub kliknięcie prawym klawiszem myszy i wybranie *Add Input* lub *Remove Input.*

string 0..n-1 **– wejścia zmiennych tekstowych , lub tablice tekstowe.**

concatenated string **– tekst wynikowy.**

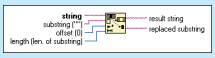


String Subset zwraca fragment tekstu o podanej długości od wybranego znaku.

string **– tekst wejściowy.**

offset **określa, od którego znaku należy rozpocząć odliczanie elementów. Gdy wejście nie jest dołączone lub wartość jest ujemna, domyślnie funkcja przyjmuje 0.**

length **określa, ile znaków z tekstu chcemy pobrać. Jeśli wejście nie zostanie dołączone, funkcja poprzyjmuje domyślnie, że jest to długość tekstu wejściowego minus wartość offset.**



Replace Substring wstawia usuwa lub zamienia tekst *substring* o określonej długości w *string*. **Jeśli** *substring* **jest pusty, z tekstu** *string* **zostanie usunięty fragment o długości** *length* **od miejsca wskazanego przez** *offset*. **W przypadku, gdy** *length* **jest równy zero,** *substring* **zostanie wstawiony w miejsce określone przez** *offset*.

string – tekst wejściowy poddawany modyfikacji.

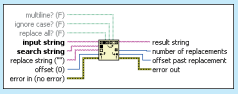
substring – tekst który zostanie wstawiony w miejsce określone przesz **offset**.

offset **– określa, od którego znaku należy wykonać wstawienie.**

length **– określa ile znaków w tekście** *string* **należy zamienić na** *substring*.

result string **– tekst wynikowy.**

replaced substring **– tekst, który został zamieniony, lub usunięty.**

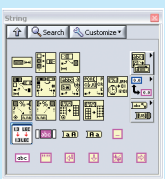


Search and Replace String zamienia, usuwa jedno lub wszystkie wystąpienia wzorca. Gdy *replace string* **jest pusty, szukany wzorzec zostaje usunięty.**

input string **– tekst wejściowy.**

search string **– szukany wzorzec.**

replace string **– tekst, w którym należy zamienić wzorzec.**



Rysunek 38. Paleta funkcji String

Tabela 1. Opis znaków formatujących, dotyczy funkcji Scan From String i Format Into String	
String	tekst nie podlegający zmianie (opcjonalnie)
%	po tym znaku następują znaki formatujące.
-	wyrównanie lewostronne brak znaku oznacza wyrównanie prawostronne (opcjonalnie)
0	dopełnienie zerami z lewej strony, brak oznacza dopełnienie spacjami (opcjonalnie)
WidthString	określa minimalną liczbę znaków (opcjonalnie)
.	oddziela liczbę określającą ilość znaków od liczby określającej precyzję
ConvChar	Znak formatujący określa typ danych d – oznacz liczbę całkowitą x - oznacza liczbę szesnastkową o – oznacza liczbę ósemkową f – oznacz liczbę stałoprzecinkową e, g – oznaczają liczby zmiennoprzecinkowe.

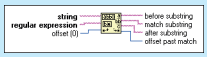
offset **– określa pozycję, od której rozpoczyna się wyszukiwanie.**

result string **– wynikowy tekst.**

number of replacements **– liczba zamienionych znaków.**

offset past replacement **– wskazuje pozycję pierwszego znaku po ostatnim zamienionym wzorcu.**

replace all? **– określa czy zamienić wszystkie, czy tylko pierwsze wystąpienie wzorca. Niepodłączenie wartości jest uznawane jako False i powoduje zastąpienie tylko pierwszego wystąpienia wzorca.**



Match Pattern to funkcja, która wyszukuje w podanym tekście wzorzec określony w *regular expression*. **Zwraca tekst przed jego wystąpieniem, wzorzec i pozostały tekst. Wskazuje pozycję pierwszego znaku w tekście po wystąpieniu wzorca. Jeśli szukany tekst nie został odnaleziony, funkcja zwraca w** *offset past match* **wartość -1, a wyjścia** *match substring*, *after substring* **są puste. Wzorzec może zawierać znaki specjalne pozwalające uogólnić wyszukanie.**

string **– przeszukiwany tekst.**

regular expression **– szukany wzorzec.**

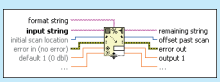
offset **– pozycja znaku, od którego należy rozpocząć przeszukiwanie.**

before substring **– tekst przed wystąpieniem wzorca.**

match substring **– wzorzec.**

after substring **– tekst po wystąpieniu wzorca.**

offset past match **– pozycja następnego znaku po znalezionym wzorcu.**



Scan From String zamienia wartości zapisane w podstacji tekstowej na właściwy typ danych, zgodnie z wzorcem formatowania. Nie znając dokładnie składni znaków formatujących można przygotować je korzystając z okna edycji. Pozwala ono na wybranie odpowiedniego formatu danych za pomocą myszki. Funkcja umożliwia jednoczesne formatowanie wielu zmiennych.

format string **– wzorzec formatowania** *[String]%[-][0][WidthString][.PrecString]ConvChar[String].*

input string **– tekst wejściowy.**

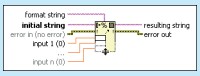
initial scan location **– pozycja, od której rozpoczyna się skanowanie.**

default 1..n **– domyślne wartości dla każdego z typów. W przypadku nieoczekiwanych danych wartości te zostaną zwrócone domyślnie.**

remaining string **– pozostały tekst.**

offset past scan **– wskazuje na następny znak po przetworzonym tekście.**

output 1..n **– wyjścia danych właściwego typu.**



Format Into String przetwarza różne typy danych do postaci znakowej. Formatowanie odbywa się zgodnie ze wzorcem formatowania, składnia opisana poniżej. Możliwe jest formatowanie więcej niż jednego typu jednocześnie. Podobnie jak w Scan From String znaki formatujące można przygotować korzystając z okna edycji.

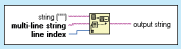
format string – wzorzec formatowania *[String]%[-][0][WidthString][.PrecString]ConvChar[String].*

initial string **– opcjonalny tekst wejściowy.**

input 1..n **– zmienne wejściowe.**

resulting string **– sformatowany tekst wynikowy.**

Znaki sterujące formatowaniem (umieszczane we wzorcu formatowania) wymieniono w **tabeli 1**.



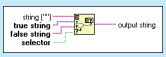
Pick Line *(funkcja z palety Additional String Functions)* – pozwala wybrać odpowiednią linie tekstu. Tekst musi być rozdzielony znakiem nowej linii.

string **– tekst wejściowy (opcjonalnie).**

multi-line string **– wybierane linie tekstu.**

line index **– numer wybranej linii.**

output string **– tekst wynikowy.**



Append True/False String *(funkcja z palety Additional String Functions)* – zależnie od wartości selector na wyjście przekazuje wartość true string lub false string.

string **– wejściowy tekst (opcjonalnie).**

true string **– tekst zwracany, gdy wartość selector jest równa** *TRUE*.

false string **– tekst zwracany, gdy wartość selector jest równa** *FALSE*.

selector **– wybór aktywnego wejścia.**

output string **– tekst wyjściowy.**

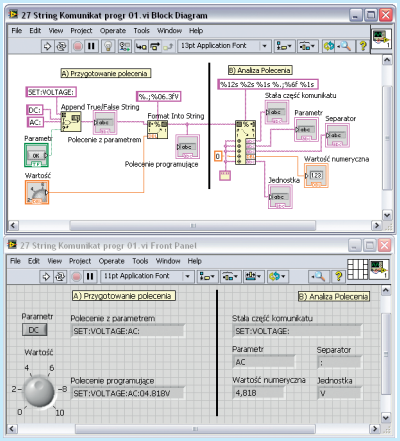
Przykład użycia funkcji z palety string

Przykładowy program ma za zadanie przygotowanie polecenia programującego przyrząd. Wymiana komunikatów pomiędzy urządzeniami pomiarowymi i komputerem odbywa się często za pomocą komunikatów tekstowych zgodnych z standardem języka SCPI. Na potrzeby przykładu założymy, że pewne urządzenie akceptuje polecenie w następującej postaci:

„SET:VOLTAGE:parametr:wartość”,

gdzie:

Parametr - przyjmuje jedną z dwóch wartości AC lub DC.



Rysunek 39. Przykład przygotowujący i analizujący polecenie programujące

wartość – jest napięciem z przedziału 0…10 V i powinna mieć postać XX.XXX. Zakładamy, że liczba określająca wartość musi mieć stała długość, część dziesiętna od całkowitej jest oddzielona kropką, stosujemy dopełnienie zerami, np. 01.220, 10.000.

Dodatkowo przyjmujemy że urządzenie odpowie takim samym komunikatem. Który należy przeanalizować i odczytać parametr, wartość numeryczną i jednostkę. Aby nie mnożyć przykładów zrobimy to na tym samym diagramie. Po analizie komunikatu powinniśmy otrzymać dokładnie takie wartości jak zadane.

Sterowanie wartością parametru zrealizujemy za pomocą przycisku. Po jego wstawieniu na panelu czołowym wybieramy *Properties* i w oknie *Appearance* zaznaczamy *Multiple string*. W polu *On text* wpisujemy *AC* a *Off text*, *DC*. W oknie *Operation* wybieramy *Switch when pressed*. Możemy nazwać kontrolkę na przykład *Parametr*. Nastawianie wartości możemy zrealizować dowolną kontrolką numeryczną np. *Knob* i nazwać ją *Wartość*. Pierwsza część komunikatu jest niezmienna, więc możemy wpisać ją w stałą tekstową. Wstawienie właściwego parametru zrealizujemy za pomocą funkcji *Append True/False String* współpracującą z naszym przyciskiem. Pozostałą część komunikatu przygotujemy korzystając z funkcji *Format Into String*. Właściwy format liczby można skonfigurować wpisując w stałą podłączoną do wejścia **format string** odpowiedni tekst formatujący lub po dwukrotnym kliknięciu myszką na funkcji ustawić go w oknie konfiguracyjnym. Po jego zamknięciu na diagramie pojawi się stała z odpowiednim tekstem formatującym.

Po połączeniu wszystkich elementów tak jak na **rysunku 39** (części A po lewej stronie diagramu) możemy przetestować fragment odpowiedzialny za przygotowanie polecenia. Pozostała jeszcze analiza komunikatu. Ponieważ jest on nieskomplikowany, to w zasadzie do jego analizy wystarczy tylko jedna, odpowiednio skonfigurowana funkcja Scan *From String*. Zauważmy, że długość komunikatu jest stała i wszystkie jego składowe zawsze zajmują te same pozycje. W pierwszej kolejności odczytamy stałą część komunikatu czyli „*SET:VOLTAGE:*”. Odpowiada za to fragment kodu formatującego *%12s* (zwraca pierwsze dwanaście znaków), następnie wartość parametru *%2s* (kolejne dwa znaki), później zbędny separator *%1s*. Teraz musimy określić, który znak oddziela część dziesiętną od całkowitej. Robimy to wpisując *%.*; następnie formatujemy wartość do postaci numerycznej *%6f*, ostatni znak informujący o jednostce *%1s*.

Gotowy program powinien wyglądać jak na **rysunku 39**. Wartości nastawiane w kontrolce *Wartość* powinny być takie same, jak otrzymywane po lewej stronie w *Wartość numeryczna*.

W praktyce możemy spotkać się z wieloma rodzajami plików. Ich struktura jest zależna od programisty. LabView udostępnia obsługę wielu typów plików binarnych, oferujących większe upakowanie, plików tekstowych, i specyficznych typów np. pliki archiwum *zip*. Funkcje do ich obsługi zgromadzone zostały w palecie *File I/O*.

Wybrane funkcje palety File I/O

Na **rysunku 40** zaprezentowano paletę funkcji *File I/O*. Zgromadzono w niej funkcje pozwalające nie tylko na zapis i odczyt plików, ale również na kopiowanie, usuwanie, kompresję itp. Skupimy się tylko na plikach.

Write to Text File zapisuje do pliku tekstowego tekst lub tablicę tekstową. Kolejne elementy tablicy zapisywane są w nowych liniach. Jeśli nie podamy ścieżki dostępu do pliku, wyświetli się nam okno dialogowe, w którym można wskazać miejsce zapisu i nazwę pliku. Trzeba pamiętać, aby dodać rozszerzenie.

prompt **– tekst wyświetlony w górnym pasku okna dialogowego (opcjonalnie).**

file **– ścieżka dostępu do pliku (opcjonalnie).**

text **– wejściowy tekst lub tablica tekstowa.**

refnum out **– wartość może być przekazana do innych funkcji wykonujących dalsze operacje na pliku.**

cancelled **– zwraca wartość** *FALSE*, **gdy prawidłowo wpisano ścieżkę dostępu do pliku lub** *TRUE*, **gdy ścieżka nie została wybrana.**

Read from Text odczytuje dane z pliku tekstowego. Korzystając ze zmiennej *count* **można określić, ile znaków chcemy odczytać. Wartość „-1” oznacza odczyt całego pliku. Funkcja – podobnie jak poprzednia – w przypadku braku ścieżki dostępu otwiera okno dialogowe umożliwiające wskazanie pliku.**

prompt **– tekst wyświetlony w górnym pasku okna dialogowego (opcjonalnie).**

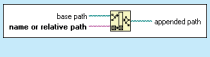
file (use dialog) **– ścieżka dostępu do pliku (opcjonalnie).**

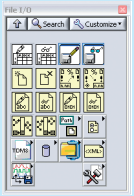
count **– określa, ile znaków chcemy odczytać z pliku.**

refnum out **– wartość może być przekazana do innych funkcji wykonujących dalsze operacje na pliku.**

text **– odczytany tekst.**

cancelled **– zwraca wartość** *FALSE*, **gdy prawidłowo wpisano ścieżkę dostępu do pliku, natomiast** *TRUE*, **gdy ścieżka nie została wybrana.**





Rysunek 40. Paleta funkcji File I/O

Build Path tworzy **ścieżkę dostępu zamieniając dane tekstowe z wejścia** *Name or relative path* **na opis ścieżki i dołączając je po ścieżce** *base path*.

base path **– podstawowa ścieżka dostępu (opcjonalnie).**

name or relative path **– nazwa pliku, folderu lub ścieżka dostępu.**

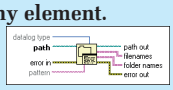
appended path **– wynikowa ścieżka dostępu.**

Strip Path wyodrębnia ze ścieżki dostępu ostatni element i zamienia go na typ znakowy.

path **– wejściowa ścieżka dostępu.**

stripped path **– ścieżka dostępu pozbawiona ostatniego elementu.**

name **– wyłuskany element.**



List Directory (z palety *Advanced File VIs and Functions*) zwraca listę plików i folderów w katalogu określonym w ścieżce path.

path – ścieżka dostępu do badanego katalogu.

pattern – wzorzec pozwalający odfiltrować nazwy np. *.txt pozwoli na zwrócenie tylko listy plików tekstowych.

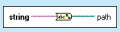
filenames – lista plików w wskazanym katalogu.

folder names – lista folderów w wskazanym katalogu.

Path To String (z palety *Advanced File VIs and Functions*) konwertuje ścieżkę dostępu do typu znakowego zachowując jej strukturę.

path **– wejściowa ścieżka dostępu.**

string **– wynik konwersji.**



String To Path (z palety *Advanced File VIs and Functions*) – konwertuje ścieżkę dostępu zapisaną w postaci tekstowej do odpowiedniego typu.

String **– ścieżka dostępu w postaci znakowej.**

Path **– ścieżka dostępu po konwersji na właściwy typ.**

Path to Array of Strings (z palety *Advanced File VIs and Functions*) – konwertuje ścieżkę dostępu do tablicy tekstowej, w której elementy są kolejnymi elementami tablicy.

path **– ścieżka dostępu.**

relative **– sygnalizuje czy jest ścieżka względna czy bezwzględna.**

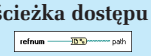
array of strings **– wynikowa tablica tekstowa.**

Array of Strings to Path (z palety *Advanced File VIs and Functions*) – konwertuje elementy tablicy tekstowej do ścieżki dostępu. Wejście *relative* **pozwala na określnie czy jest to ścieżka względna, czy bezwzględna.**

relative **– wartość** *FALSE* **jest domyślną i oznacza, że ścieżka dostępu jest bezwzględna, natomiast** *TRUE* **oznacza ścieżkę względną.**

array of strings **– tablica tekstowa zawierająca kolejne elementy ścieżki dostępu.**

path **– wynikowa ścieżka dostępu właściwego typu.**



Refnum to Path (z palety *Advanced File VIs and Functions*) – konwertuje typ *refnum* do odpowiedniej ścieżki dostępu.

refnum – jest wskazaniem na otwarty plik.

path **– ścieżka dostępu związana z** *refnum*.

Current VI's Path (z palety *File Constants*) – zwraca ścieżkę dostępu do pliku w którym jest umieszczona.

Przykład obsługi plików tekstowych

Przygotujemy program wykorzystujący funkcje z palety File I/O, jego zadaniem jest.

Zapis do pliku tekstowego dowolnego tekstu z dodanym na początku numerem linii.

Utworzenie pliku tekstowego w tym samym katalogu, co program.

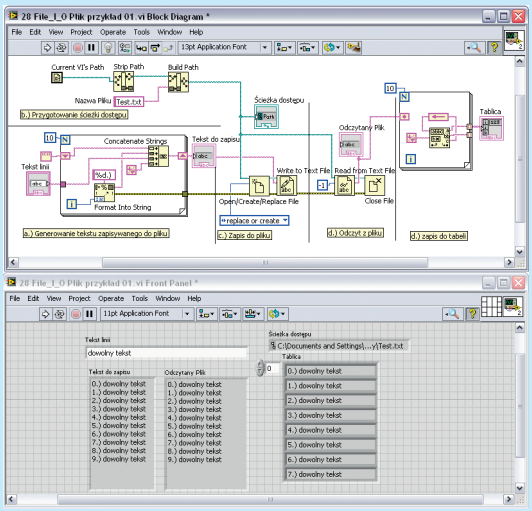
Odczytanie zawartości pliku.

Podzielenie odczytanego tekstu na linie i umieszczenie ich w kolejnych komórkach tablicy.

Punkty 1 i 4 zostały dodane w celu przećwiczenia operacji na zmiennych tekstowych. Można je pominąć i wykonać tylko czynności z punktów 2 i 3.

Tworzenie programu rozpoczynamy od przygotowania tekstu zapisywanego do pliku. Dla uproszczenia przyjąłem, że zawartość każdej z linii będzie taka sama, a na jej początku zostanie dodany numer. Liczba linii też jest stała i określona liczbą iteracji pętli. Tekst zawarty w każdej z linii jest pobierany z kontrolki *Tekst*. W pętli *For* funkcja *Format Into String* zamienia numer iteracji na zmienną tekstową, która jest dopisywana na początku każdej z linii za pomocą *Concatenate Strings*. Na końcu jest dopisany znak końca linii. Gotowy do zapisu tekst można zobaczyć we wskaźniku *Tekst do Zapisu*.

Plik zapisany zostanie w tym samym katalogu, co nasz program. Korzystając z *Current VI's Path* określamy, gdzie znajduje się program. Otrzymujemy **ścieżkę dostępu do programu**. **UWAGA! Program przed uruchomieniem należy zapisać na dysku, ponieważ Current VI's Path nie zwróci ścieżki dostępu**. Za pomocą *Strip Path* odcinamy nazwę programu. Funkcją *Build Path* dodajemy nazwę pliku tekstowego wraz z rozszerzeniem.



Rysunek 41. Przykład operacji na plikach

Możemy wygenerować na panelu czołowym wskaźnik wyświetlający miejsce zapisu pliku. Ja nazwałem go **Ścieżka dostępu.**

Przed zapisem należy korzystając z funkcji *Open/Create/Replace File* określić sposób edycji jako *replace or create*, gdyż w przeciwnym wypadku każdy zapis do pliku będzie skutkował dopisaniem nowego tekstu na końcu istniejącego. Sam zapis realizuje *Write to Text File*. Klikając na niej prawym przyciskiem myszy w menu lokalnym trzeba odznaczyć opcję *Convert EOL*.

Odczyt z pliku realizuje *Read from Text File*, wartość -1 na wejściu *count* oznacza, że czytamy całą zawartość pliku. Tutaj również należy odznaczyć *Convert EOL*.

Odczytany tekst jest przekazany do pętli, **w której** *Match Pattern* wyszukuje znak końca linii i poprzedzającą go część przesyła do tunelu wyjściowego pętli. Pozostały tekst jest przesyłany do *Feedback Node* i analizowany w kolejnej iteracji. Cały **program przedstawia rysunek 41**.

Zasady tworzenia wykresów

Jednym z elementów przetwarzania danych jest ich wizualizacja. Najłatwiejszym i najbardziej czytelnym sposobem jest przedstawienie ich w postaci wykresów. LabView daje możliwość przedstawienia danych na wykresach przebiegów, XY trójwymiarowych, intensywności, wykresach kołowych i innych. Przedstawię tutaj zasady tworzenia tylko podstawowych i najczęściej używanych wykresów.

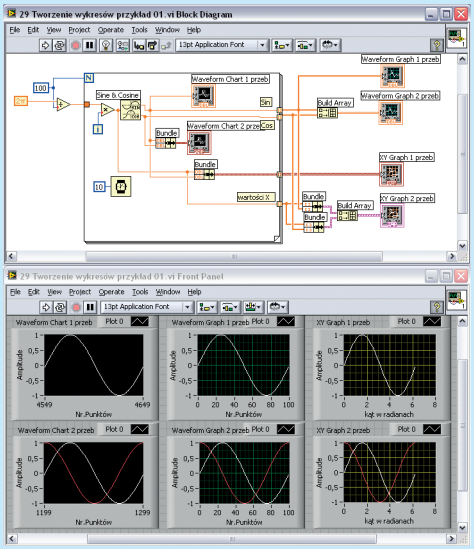
Wszystkie wykresy zgromadzone zostały na panelu czołowym w palecie *Graph*. Po umieszczeniu w oknie panelu odpowiedniego wykresu należy przejść na diagram i wykonać połączenie zacisku ze źródłem danych o odpowiednim formacje. Format danych zależy od typu wykresu i sposobu wykreślania, co zostało opisane poniżej.

Wykres Waveform Chart

Waveform Chart jest najprostszym z wykresów. Służy do wyświetlania aktualnie zbieranych danych, wejście akceptuje wartość skalarne które w miarę napływania dorysowywane są na wykresie i gromadzone w wewnętrznej tablicy. Wykreślany może być jeden lub wiele przebiegów jednocześnie. Zależnie od typu dostarczanych danych. W **tabeli 2** przedstawiono sposób wyświetlania przebiegów w zależności od typu danych.

Wykres akceptuje też specjalny typ danych reprezentujący przebiegi czasowe *Waveform Data Type* w skrócie *WDT*. Struktura jest podobna do klastra danych zawiera informację o czasie początkowym przebiegu (x0), odstępie czasowym pomiędzy próbkami (dt) i tablicę próbek (Y).

Na osi x może znajdować się czas lub numery próbek. Każdy z wykresów pozwala na konfigurowanie wielu opcji większość z nich jest dostępna w menu lokalnym, i taka sama dla części wykresów są to np. kolory przebiegów styl linii, grubość linii, rodzaj punktów, automatyczna zmiana skali oddzielnie dla każdej osi, typ osi liniowy logarytmiczny i wiele innych. Dostęp do wszystkich tych opcji jest możliwy również w sposób programowy. Na szczegółowy opis wszystkich opcji nie ma tutaj miejsca.



Rysunek 42. Przykłady tworzenia wykresów

Dlatego skupię się tylko na odpowiednim przygotowaniu danych. Przykłady tworzenia wykresów *Waveform Chart* pokazano na **rysunku 42**.

Wykres - Waveform Graph

Waveform Graph, podobnie jak *Waveform Chart*, prezentuje przebiegi, w których na osi poziomej znajduje się numer próbki lub czas. Różnica polega na tym, że tutaj od razu jest wyświetlany cały przebieg. Do zacisku wejściowego należy przesłać tablicę zawierającą wszystkie punkty przebiegu. Możliwe jest wyświetlenie jednego lub więcej przebiegów w jednym oknie. **Tabela 3** opisuje sposób wyświetlania w zależności od typu danych. Na **rysunku 42** pokazano przykłady tworzenia wykresów *Waveform Graph* wyświetlających pojedynczy i podwójny przebieg.

Wykres – XY Graph

XY Graph przedstawia wartości funkcji w kartezjańskim układzie współrzędnych. Dane wejściowe zawierają dwie tablice zgrupowane w klaster danych, pierwsza zawiera wartości dla osi X a druga dla osi Y. Podobnie jak na poprzednich wykresach można wyświetlić jednocześnie więcej niż jedną funkcję wówczas klastry danych należy zgrupować w tablicę. Przykład pokazano na **rysunku 42**.

Tabela 2. Sposób wyświetlania przebiegów w zależności od typu danych

Rodzaj danych	Sposób wyświetlania
Skalar	1 przebieg - 1 punkt
Tablica 1D	1 przebieg - 1 lub więcej punktów
Tablica 2D	kilka przebiegów – 1 lub więcej punktów
Klaster danych	kilka przebiegów – 1 lub więcej punktów
WDT	1 przebieg - 1 lub więcej punktów

Tabela 3. Sposób wyświetlania w zależności od typu danych

Rodzaj danych	Sposób wyświetlania
Tablica 1D	1 przebieg
Tablica 2D	kilka przebiegów w jednym oknie
WDT	1 przebieg

Przykłady tworzenia wykresów

Aby pokazać jak przygotować dane dla poszczególnych wykresów, zrobimy program liczący funkcje sinus i cosinus oraz przedstawimy je na powyżej scharakteryzowanych wykresach. Funkcje trygonometryczne operują na mierze łukowej dlatego korzystamy z radianów a nie stopni. Na początku wyznaczamy krok a następnie w każdej z iteracji obliczamy wartość funkcji dla kolejnych wartości kąta. Wykorzystaną tutaj funkcję *Sine & Cosine* znajdziemy w palecie *Mathematic → Elementary & Special Functions → Trigonometric Functions*. Proponuje dodać opóźnienie czasowe, aby zobaczyć jak jest wykreślany przebieg *Waveform Chart*. Dla każdego z wykresów przedstawiono dwa przypadki, gdy jest wyświetlany jeden przebieg oraz dwa przebiegi.

Wykresy *Waveform Chart* znajdują się wewnątrz pętli, podczas każdego jej obiegu dorysowują jedną wartość. Aby uzyskać tylko jeden przebieg wystarczy przekazać wartość wprost z funkcji, dla uzyskania dwóch wykresów wartości funkcji sinus i cosinus zostały połączone w klaster danych przekazany do *Waveform Chart 2 przeb*.

Podobnie wygląda sytuacja z *Waveform Graph*. Do uzyskania jednego przebiegu wystarczy przekazanie tablicy jednowymiarowej. Natomiast, aby wyświetlić dwa przebiegi, trzeba przekazać tablicę dwuwymiarową. Wykresy umieszczone są na zewnątrz pętli, gdzie dostępne są wszystkie wartości.

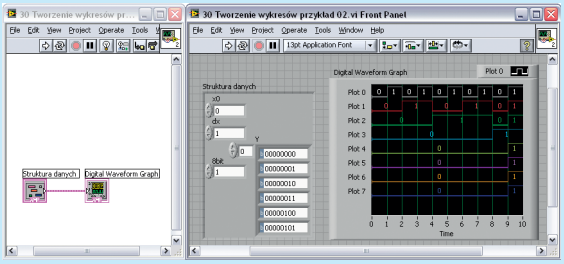
Również wykres XY Graph musi być umieszczony poza pętlą, ponieważ dane dla każdej z osi muszą być zgromadzone w tablicach. Zwróćmy uwagę na sposób grupowania danych dla *XY Graph 1 przeb*. Zostały one zgrupowane w klaster wewnątrz pętli i w ten sposób uzyskaliśmy tablicę klastrów będących parami wartości dla osi X oraz Y. Właściwe jest również zgrupowanie danych poza pętlą – wówczas otrzymamy klaster danych zawierający dwie tablice przechowujące wartości X i Y. W obu przypadkach przebiegi zostaną wyświetlone poprawnie.

Wykres - Digital Waveform Graph

Digital Waveform Graph jest odmiennym rodzajem wykresu. Jest przeznaczony do wyświetlania bitowej reprezentacji cyfrowych przebiegów. Dane wejściowe mają postać klastra danych. Na pierwszej pozycji znajduje się liczba opisująca czas początkowy *x0*, odstęp czasowy pomiędzy próbkami *dx*, tablicę danych, liczbę opisującą szerokość danych. Przyjęto że podstawowa szerokość wynosi 8 bitów. Dla liczby 8-bitowej podajemy 1, 16-bitowej 2 itd. **Rysunek 43** przedstawia przykładowe przebiegi.

Sterowanie właściwościami elementów panelu czołowego

Property Node daje możliwość sterowania właściwościami obiektów panelu czołowego. Mogą one być zarówno zapisywane, jak i odczytywane. Aby wstawić *Property Node* na diagramie należy z menu lokalnego danego elementu wybrać *Create → Property Node → Value* (odpowiedni atrybut na przykład Value). Zacisk możemy rozciągać za pomocą myszki, tak aby mieć dostęp jednocześnie do większej liczby atrybutów. Mogą być one mieszane, czyli jednocześnie możemy część z nich zapisywać, a część odczytywać. Najeżdżając



Rysunek 43. Przykładowy wykres Digital Waveform Graph

myszką na wybrany atrybut, po pojawieniu się „rączki" i kliknięciu, rozwinie się lista dostępnych atrybutów, z której wybieramy odpowiedni. Domyślnie większość ma zacisk z prawej strony pozwalający na odczytywanie właściwości. Klikając prawym klawiszem myszki z menu lokalnego należy wybrać *Change To Write*, gdy chcemy zapisywać wartość atrybutu. Wówczas zacisk pojawi się z lewej strony. Zmiana odwrotna jest możliwa po wybraniu *Change To Read*. Takiej samej zmiany można dokonać jednocześnie dla całej grupy atrybutów wybierając odpowiednio *Change All To Write* albo *Change All To Read*. Na **rysunku 44** pokazano przykładowe zaciski dla dwóch różnych kontrolek wykresu i przycisku. Część właściwości jest taka sama np. *Value* (wartość), *Visible* (widoczność elementu). Część jest specyficzna dla danej kontrolki np. *PlotArea Size* (wielkość obszaru aktywnego wykresu), część jak *Val(sgnl)* może być tylko zapisywana.

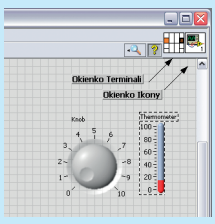
Wszystkich właściwości jest zbyt dużo, aby je opisać. Myślę że ich nazwy są dość czytelne i sugerują czego dotyczy dany atrybut. Ponadto, klikając prawym przyciskiem myszy na wybranym atrybucie, w rozwiniętym menu lokalnym znajdziemy opcję pozwalającą na otwarcie okno pomocy danej właściwości np. *Help For Value*. Są tam wszystkie niezbędne informacje.

Przykład sterowana oknem wykresu Waveform Graph

Przykładowy program wyświetla funkcję sinus i kosinus na wykresie *Waveform Graph*. Korzystając z *Property Node* określimy: nazwy przebiegów, styl punktów jednego z przebiegów, nazwę osi X oraz kolor tła.

Diagram odpowiedniej funkcji przedstawia **rysunek 45**. Obliczenie funkcji sinus i cosinus zrealizowano tak samo, jak na przykładzie z **rysunku 42**. Wynik został wyświetlony na wykresie *Waveform Graph*. Klikając na ikonie wykresu należy z menu lokalnego wybrać *Create → Property Node → Active Plot*.

Na diagram zostanie wstawiony zacisk z pierwszym atrybutem. Korzystając z myszki należy rozciągnąć go tak, aby widoczne było siedem atrybutów. Klikając myszką na każdym z nich, gdy jest widoczna „rączka" należy z listy wybrać odpowiednie atrybuty zgodnie z kolejnością widoczną na **rysunku 45**. Niektóre z nich zostały pogrupowane zgodzie z elementem, którego dotyczą np. *Plot.Name* oznacza, że należy szu-



Rysunek 44. Przykładowa lista właściwości wykresu Waveform Chart i przycisku Boolean

kać w grupie *Plot* pod nazwę *Name*. Kropli widoczne w nazwach właściwości oznaczają kolejne grupy. Jeśli terminale połączeniowe znajdują się z prawej strony, należy z menu lokalnego wybrać *Change All To Write* i połączyć wszystko, jak pokazano na **rysunku 45**. Kontrolka *Plot Area:Colors:BG Color* pozwala na wybranie kolor z palety barw za pomocą myszki. Znajduje się w palecie *Numeric* pod nazwą *Framed Color Box*. Można ją zastąpić zwykłą zmienną numeryczną, ale określenie koloru będzie trudne.

W każdej iteracji pętli rysowany jest wykres i ustawiane wybrane właściwości *WaveForm Graph*.

Property node wykonuje się następująco:

Active Plot=0 – ustawia pierwszy wykres aktywnym, wszystkie właściwości dotyczące przebiegu będą dotyczyły właśnie pierwszego wykresu.

Plot Name=Sin – określa nazwę pierwszego przebiegu.

Active Plot=1 – drugi wykres jest aktywny.

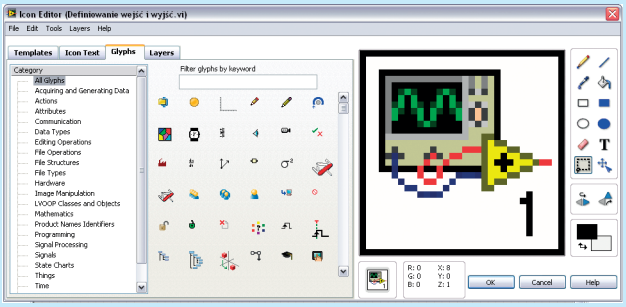
Plot Name=Cos – określa nazwę drugiego przebiegu.

Point Style =1 – określa rodzaj punktów wykresu.

XScale.NameLbl.Text = Kąt – ustawia opis osi X.

BG Color – określa kolor tła okna wykresu.

Oczywiście umieszczenie *Property Node* bezpośrednio w pętli nie jest dobrym rozwiązaniem. Najlepiej wywoływać je tylko wtedy, gdy zachodzi konieczność zmiany lub sprawdzenia określonych właściwości. W przeciwnym razie niepotrzebnie zajmuje czas procesora.



Rysunek 45. Przykład zastosowania Property Node do sterowania właściwościami wykresu

Podsumowanie

W tej części poznaliśmy operacje na danych tekstowych, plikach, zasady tworzenia wykresów oraz sterowanie właściwościami elementów panelu czołowego. Do tej pory przygotowywane przez nas programy mieściły się na jednym diagramie. W kolejnej części nauczymy się pracy z menedżerem projektów, definiowania wejść i wyjść funkcji. Napiszemy prostą grę bitwa morska przechodząc wszystkie etapy tworzenia aplikacji, aż po wygenerowanie pliku exe.

Wiesław Szaj
wszaj@prz.edu.pl