Zmienne współdzielone

Podczas dzisiejszego ćwiczenia będziemy uruchamiać zadania na komputerach docelowych z zainstalowanym system czasu rzeczywistego ETS firmy National Instruments. Poznamy zmienne umożliwiające komunikację pomiędzy aplikacjami działającymi na jednym komputerze lub na różnych komputerach, zwane zmiennymi współdzielonymi (*shared variables*). Na koniec nauczymy się, jak pisać do pliku tekstowego (ASCII).

1. Zapoznanie się ze zmiennymi współdzielonymi

Zanim zaczniemy programować nowe zadanie należy obejrzeć krótki film **shared_var.swf**. Dotyczy on starszej wersji programu (LabVIEW 8) i innej konfiguracji systemu, ale również zawiera kasetę PXI i zastosowane metody tworzenia zmiennych są prawie identyczne.
Proszę nie wykonywać oglądanych czynności, a jedynie obejrzeć film z uwagą. Proszę zwrócić uwagę, że zmienne te są dużo bezpieczniejsze od zmiennych lokalnych, ze względu na możliwość buforowania strumieni danych.

2. Zmienne współdzielone działające w ramach jednego komputera

Proszę utworzyć nowy projekt **Nazwisko_proj_9** zawierający komputer lokalny (My Computer) oraz komputer docelowy PXI. Do projektu dołączmy dwa przygotowane przeze mnie programy: **Host_9.vi** ma działać na komputerze głównym, a **Target_9.vi** na komputerze docelowym.
Na razie będziemy się zajmować jedynie programem **Target_9.vi,** gdzie umieszczone są dwie pętle czasowe: jedna, o nazwie **P200**, i o wyższym priorytecie (równym 200), a druga, **P50**, o priorytecie równym 50. Proszę zwrócić uwagę, że priorytety, nazwy pętli oraz źródła czasu są zdefiniowane nie przez podpięcie stałych do wejść pętli czasowych, lecz poprzez ustawienie parametrów, czego można dokonać po kliknięciu prawym klawiszem myszy w panel wejść i wybranie opcji ***Configure Input Node***.
W pętli o wyższym priorytecie (200) umieszczony jest generator fali sinusoidalnej (można go znaleźć w zakładce *Functions>Waveform>Analog Waveform>Generation*) – proszę zapoznać się ze zdefiniowanymi parametrami generacji sygnału. Natomiast w pętli P50 jest ikonka wyświetlacza graficznego *Waveform Graph*. Przekażemy dane, czyli generowany przez generator sygnał, z jednej pętli do drugiej za pomocą zmiennej współdzielonej. W tym celu należy kliknąć prawym przyciskiem myszy w symbol **komputera docelowego** na oknie projektu i wybrać *New>Variable*. Pojawi się okno, gdzie nadajemy nazwę (na przykład **DaneGen**), *Variable Type = Single Process*, *Data Type = Double Waveform* i, na razie, nic więcej. Zmienna pojawiła się w projekcie w zakładce *Untitled Library 1*, której nazwę zmieniamy poprzez zapisanie jej na dysku jako np. **Zmienne_9**. Przeciągamy tę zmienną do pętli z generatorem. Umożliwiamy pisanie do tej zmiennej przez kliknięcie w bloczek zmiennej i wybór *Acces Mode> Write* i łączymy ją z wyjściem z generatora. Analogicznie umieszczamy tę zmienną w pętli z wyświetlaczem *Waveform Graph* i łączymy z wejściem tego wyświetlacza. Ten program nie potrafi się jeszcze zatrzymać, więc analogicznie, tworzymy zmienną współdzieloną o nazwie np. STOP, z tym, że ma ona być zmienną logiczną. **UWAGA**: by nie robić kolejnej biblioteki ze zmiennymi NIE klikamy teraz w ikonę komputera docelowego (w oknie projektu), lecz w ikonę biblioteki **Zmienne_9.lvlib** i wybieramy *New> Variable*. Nowa zmienna utworzy się w tej samej bibliotece, co poprzednia. Podłączamy nową zmienną tak, by naciśnięcie przycisku STOP zatrzymało obie pętle. Aby nie było problemu z inicjalizacją, dobrze jest na lewo od pętli umieścić tę zmienną

z podpiętą wartością FALSE. Należy wyjście błędu zmiennej podpiąć do wejścia błędu pętli P200, wtedy na pewno inicjalizacja zmiennej nastąpi przed uruchomieniem pętli. Program należy uruchomić i zaobserwować sinusoidalny przebieg na ekranie wyświetlacza. Gdy klikniemy STOP, to program się zatrzyma.

3. Buforowanie danych

Zmienna **DaneGen** umożliwia buforowanie danych. Zauważmy, że zgodnie z podanymi parametrami na wejściu *sampling info* generatora[1], generator generuje za jednym razem 1000 punktów oraz generuje 1000 punktów na sekundę tzn., że raz na sekundę generowana jest paczka 1000 punktów – za punkt rozumiem tutaj parę: czas - wartość sygnału. Będziemy buforować te dane. Klikając dwukrotnie w oknie projektu w zmienną **DaneGen** otwieramy okno z własnościami zmiennej i przechodzimy do zakładki **RT FIFO**, klikamy *Enable RT FIFO* i wybieramy ilość punktów na sygnał. Proszę zobaczyć jak program będzie działał, jeżeli wybierzemy 1, 100, 1000, 1100 lub 1500 punktów. *FIFO type* pozostaje pojedynczym elementem, gdyż przesyłamy jeden sygnał, a nie ich tablicę. Gdy bufor jest za mały, to tylko część sygnału jest przesyłana, a gdy za duży, to może się zdarzyć przesłanie niezainicjalizowanego fragmentu pamięci (widzimy szum). Ostatecznie ustawiamy 1000. Nie włączamy żadnego buforowania danych dla zmiennej logicznej STOP, co jest związane z tym, że ta zmienna jest z definicji chwilowa w czasie. Buforowanie będzie powodować zatrzymywanie się programu po jego ponownym uruchomieniu, gdyż pojawiać się mogą nieusunięte elementy z bufora.

4. Reżym czasu rzeczywistego - przesyłanie danych pomiędzy dwoma komputerami

Zrobimy teraz z tego programu program działający (prawie) w reżymie czasu rzeczywistego. W tym celu wyrzucamy z programu **Target_9.vi** bezpośrednie oddziaływanie z użytkownikiem, czyli przycisk STOP i wyświetlacz *Waveform Graph*. Zmieniamy zmienną STOP, by była typu **Network-Published** oraz tworzymy w projekcie nową współdzieloną zmienną **DaneGenNet**, o identycznych własnościach jak **DaneGen**, lecz typu **Network-Published** - dobrze jest zdefiniować buforowanie nie tylko w zakładce RT FIFO, ale i Network. Nowe zmienne podłączamy tak, by sygnał **STOP** przekazywać z **Host_9.vi** do **Target_9.vi**, gdzie ma zatrzymywać obie pętle, a sygnał generatora z **Target_9.vi** na wyświetlacz *Waveform Graph* w **Host_9.vi**. W **Target_9.vi** podłączenia zmiennej sieciowej dokonujemy w pętli o mniejszym priorytecie.
Inicjalizacji zmiennej **STOP** dokonamy w nietypowy sposób, by uniezależnić się od tego, który program zostanie uruchomiony najpierw. A mianowicie, na komputerze docelowym umieszczamy zmienną **STOP** na końcu programu w ten sposób, by wpisanie w nią wartości FALSE było ostatnią operacją wykonaną przed końcem działania programu.
Uruchamiamy oba programy i sprawdzamy działanie systemu.

5. Zapisywanie w pliku tekstowym daty i godziny.

W zakładce *Functions>Programming>File I/O* znajdują się funkcje służące do operacji na plikach. Zapoznamy się z wybranymi funkcjami oglądając przykład. W tym celu należy w Menu Help wybrać *Help> Find Examples* i w oknie, które się otworzy, wybrać podwójnym kliknięciem

[1] Jest tam podłączona tablica dwóch liczb równych 1000 – pierwsza, to ilość próbek (punktów) generowanych na sekundę (częstość generacji), a druga, to ilość generowanych punktów za jednym wywołaniem funkcji generatora (w jednej paczce).

Fundamentals, następnie **File Input and Output**, następnie **Text (ASCII)**, po czym otwieramy program **Write to Text File and Read from Text File.vi**. Należy go obejrzeć.

Operacja pisania do pliku umieścimy w programie **Host_9.vi**. W tym celu proszę stworzyć w tym pliku nową, zwykłą pętlę *while*, która będzie się wykonywać raz na sekundę. Korzystając z funkcji *Build Path*, *Open/Create/Replace File*, *Write Text File* oraz *Close File* należy zapisywać do zbioru aktualny czas i datę. Aby utworzony plik znajdował się w tym samym miejscu, co program vi, należy użyć stałej *Current VI's Path* oraz funkcji *Strip Path* w następujący sposób: ze stałej należy wpierw wydzielić samą ścieżkę do katalogu (za pomocą *Strip Path*), a następnie dodajemy do ścieżki naszą nazwę pliku z zapisaną datą (np. czas.txt) za pomocą funkcji *Build Path*. Plik otwieramy z parametrami zamień lub utwórz (**replace or create**) oraz tylko do pisania (**write-only**). Aby to sobie ułatwić najlepiej kliknąć w odpowiednie wejścia funkcji *Open/Create/ Replace File* i wybrać Create/Constant – utworzy się stała zawierająca wszystkie możliwości.

Napis zawierający aktualny czas ma być zapisywany do pliku za każdym obrotem pętli *while*. Umieszczamy w pętli funkcję *Wait Until Next ms Multiple* z opóźnieniem 1000 ms, by zbiór czas.txt nie zrobił się zbyt duży. Napis zawierający aktualny czas tworzymy korzystając z funkcji ***Get Date/Time String*** oraz ***Concatenate Strings***. Format ma wyglądać tak (czas zawiera sekundy):

27 maja 2015, 17:19:08

Każdy taki zapis ma znajdować się w nowej linii.

Aby program działał poprawnie, potrzebna będzie jeszcze jedna zmienna typu *Single Process,* która umożliwi zatrzymywanie pętli piszącej datę i czas.