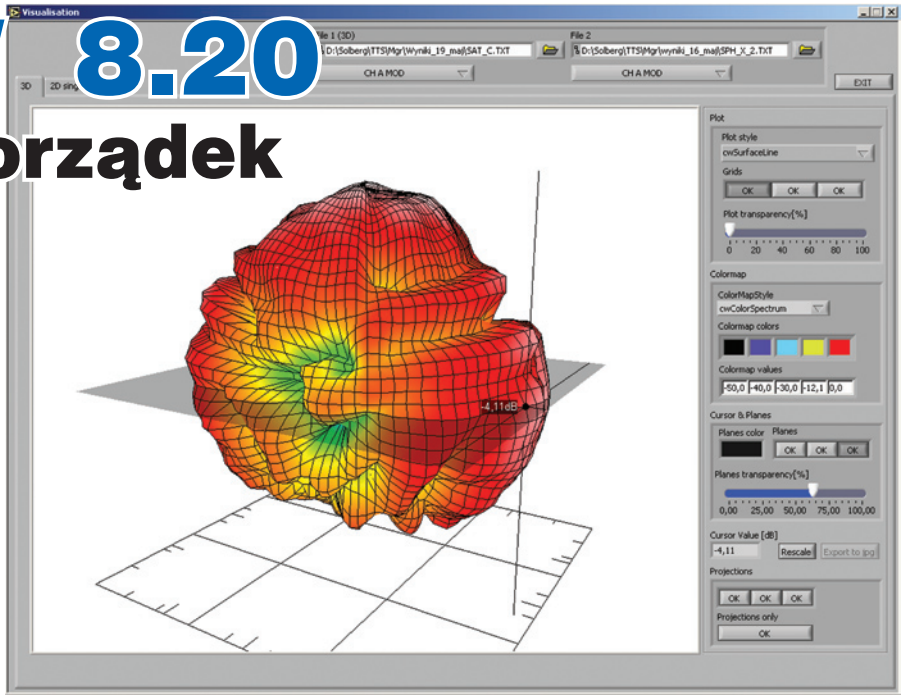
LabVIEW 8.20
Globalny porządek



Trudno w to dzisiaj uwierzyć, ale pierwsza wersja pakietu LabVIEW (oferowana wtedy wyłącznie w wersji na komputery Apple Macintosh) ujrzała światło dzienne w 1986 roku, a co za tym idzie, wprowadzona w życie przez National Instruments idea programowania graficznego ma już 20 lat.

Wspomniana rocznica jest okazją do uzmysłowienia sobie, że podczas ówczesnej komputerowej rewolucji (kiedy to PC trafiały pod „strzechy"), zaledwie po 8 latach od powstania specyfikacji języka C, twórcy LV szukali nowych, przystępnych dla szerokiego grona użytkowników sposobów tworzenia programów. Szukano sposobu na zastąpienie „surowych" tekstowych programów, stosowanych w tradycyjnych językach programowania.

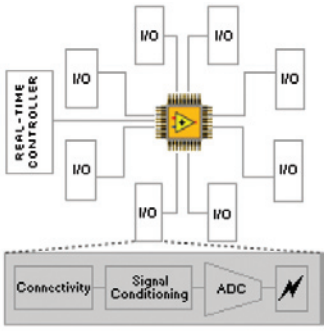
Dlaczego uważam ideę programowania graficznego za przystęp-

Rys. 1. Wygląd kontrolera CompactRIO

niejszą? – zapytają sceptycy (tradycjonaliści). Jedną z najbardziej banalnych odpowiedzi może być powiedzenie: Jeden obrazek jest często więcej wart niż 1000 słów…

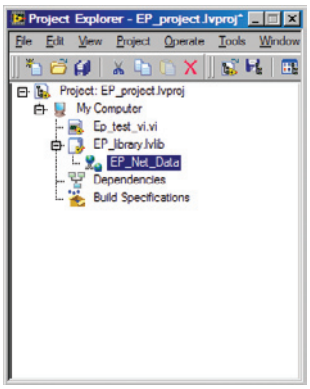
Z okazji 20–lecia wdrożenia jednego ze sztandarowych produktów firmy National Instruments, wprowadzono na rynek specjalną edycję LabVIEW 8 nazwaną 8.20 (liczba dwadzieścia nawiązuje do rocznicy). Wersja ta gromadzi wszystkie nowe funkcjonalności pakietu LV, jakie zostały wprowadzone do wersji 8, jak również dotychczasowe moduły dla różnych platform sprzętowych. Dwie największe innowacje wersji 8 (i naturalnie 8.20) to *Explorer Projektu* oraz *Zmienne Współdzielone*. Przypomnę pokrótce ich znaczenie (w wersji 8), co pozwoli szybko i obiektywnie ocenić zmiany wprowadzone w wersji 8.20.

Począwszy od przedstawienia przez National Instruments kontrolerów czasu rzeczywistego oraz oprogramowania dla nich (moduł LV Real–Time), pojawiał się początkowo niedostrzegany problem integracji aplikacji pracujących w łańcuchu platform sprzętowych. Typowa architektura to komputer (czy też kontroler np. CompactRIO) RT odpowiedzialny za bezpośrednią kontrolę procesów, bądź gromadzenie szybkozmiennych danych oraz komputer klasy PC, prezentujący i archiwizujący te dane. Oba węzły połączone są ze sobą siecią Ethernet i komunikują się zdefiniowanym protokołem. Moduł CompactRIO (**rys. 1**) to tak naprawdę kontroler RT z systemem operacyjnym oraz *backplane* (rodzaj elektryczno–mechanicznej bazy) modułów I/O kontrolowany przez odpowiednio skonfigurowany układ FPGA (**rys. 2**)



Rys. 2. Architektura systemu stosowana w CompactRIO

Wszystko to po to, aby zapewnić jak największą możliwą skalowalność systemu, umożliwiającą jednocześnie dostęp do zasobów sprzętowych na niskim poziomie (np. możliwość konstruowania pętli pracujących z maksymalną częstotliwością 40 MHz) z warstwy tworzonej aplikacji. Wszystko to brzmi bardzo obiecująco, jednakże należy pamiętać, iż do obsługi całego łańcucha potrzebujemy nie tylko podstawowego pakietu LV, ale również modułu LabVIEW RT oraz LabVIEW FPGA. To jednak nie koniec. Musimy stworzyć rozproszoną aplikację, której fragmenty będą osadzone na poszczególnych węzłach systemu. Pomimo, iż środowisko do programowania jest skupione na jednym komputerze PC, to tak naprawdę będziemy tworzyć trzy oddzielne aplikacje, za każdym razem używając innego zestawu funkcji pochodzących z różnych modułów. O ile architektura i implementacja takiej aplikacji nie nastręcza tak wielu problemów, to problem pojawia się w momencie jej uruchamiania i testowania. Okazuje się nagle, że niektóre fragmenty kodu, a co za tym idzie, funkcjonalności systemu, warto przenieść na inny węzeł. Zaczyna się żmudna optymalizacja z ciągłym przełączaniem platformy docelowej (*execution target*) w LabVIEW.



Rys. 3. Widok okna *Project Explorera*

Te właśnie niuanse wymusiły na National Instruments stworzenie narzędzia, które jest w stanie zapanować nad obmyśloną przez nas architekturą.

W tym celu powstał Project Explorer (**rys. 3**). Pozwala on na grupowanie plików źródłowych LV oraz innych zasobów, jak np. dokumentacji czy kodów źródłowych pisanych w innych językach niż LV, w logiczne foldery przypisane z kolei do różnych węzłów tworzonego systemu.

Doszukując się analogii do „tradycyjnych" języków programowania i tworzonych z myślą o nich IDE, możemy stwierdzić, że w każdym z nich znajduje się manager projektu: na przykład w środowisku Keil µVision nosi on nazwę *Project Workspace*, w środowisku TIDE – po prostu *Project*. Jest to jedna z głównych innowacji pakietu LV 8 w stosunku do wersji poprzedniej (a więc 7.1).

Wracając do rozpatrywanego systemu: kolejnym problemem, któremu musimy stawić czoła, to ogólny problem komunikacji pomiędzy węzłami, w kolejnym przybliżeniu dotykający problemu synchronizacji pracy. Nic nie stoi na przeszkodzie opracowania protokołów komunikacyjnych w tradycyjny sposób: wybieramy warstwę transportową (np. UDP) i implementujemy na niej potrzebne sekwencje rozkazów i odpowiedzi. Podejście to jakkolwiek prawidłowe i bezpieczne, jest

Rys. 4. Symbol zmiennej współdzielonej

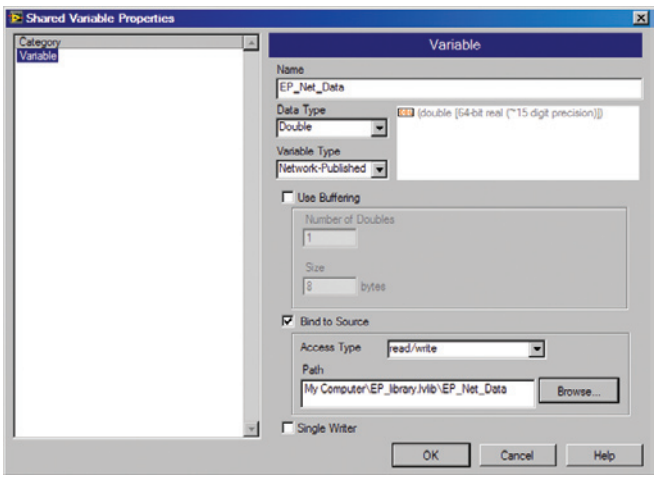
uciążliwe w implementacji oraz testowaniu. Aby abstrahować nieco od fizycznego rozdzielenia platform sprzętowych oraz od łączących je sieci, NI wprowadził tzw. zmienne współdzielone (*shared variables*) – **rys. 4**. W dużym skrócie można by je porównać do zmiennych globalnych w znaczeniu całego systemu rozproszonego.

Przy tworzeniu nowej zmiennej współdzielonej, na etapie jej konfiguracji (**rys. 5**) musimy określić jej nazwę, typ danej (w naszym przypadku *double*), oraz typ zmiennej, który decyduje o tym, czy będzie ona używana lokalnie czy dystrybuowana poprzez sieć.

Po prawidłowej konfiguracji możemy takiej zmiennej używać do zapisu/odczytu danych w każdym miejscu kodu należącego do projektu, a co za tym idzie, kodu przypisanego różnym platformom. Obydwa te mechanizmy są częścią filozofii realizacji projektów, wprowadzonej przez NI w wersji 8 LabVIEW nazwanej *Distributed Intelligence* (rozproszona inteligencja).

Co nowego w *LabVIEW 8.dwadzieścia*

W jubileuszowej wersji LV, producent zaimplementował pewne funkcjonalności rozszerzające możliwości i łatwość użycia mechanizmów rozproszonej inteligencji. Pierwszą z nich jest *FPGA Wizard* (a więc kreator FPGA). Jest to rozszerzenie modułu LabVIEW FPGA pozwalające na generowanie szablonu kodu osadzonego na układzie FPGA (czy to na karcie PCI, PXI czy w module CompactRIO). Kreator pozwala na definiowanie struktury kodu – zwykle są to dwie pętle równoległe – pierwsza, pracująca z maksymalną szybkością, odpowiedzialna za zbieranie danych z modułów pomiarowych (tak cyfrowych jak i analogowych) oraz druga, odpowiedzialna za komunikację z resztą systemu. Kreator ten pozwala niejako zaprojektować szablon programu, przy czym określamy w nim jedynie funkcjonalność kodu. W tradycyjnym podejściu, na tym etapie pracy, należałoby zacząć implementować w LV (a więc rozpocząć programowanie) po kolei wszystkie elementy systemu. Po zdefiniowaniu naszych wymagań wystarczy kliknąć przycisk: *Generate Code* (wygeneruj kod),



Rys. 5. Okno konfiguracji właściwości zmiennej współdzielonej

co spowoduje stworzenie szablonu aplikacji tak dla układu FPGA, jak i dla komputera nadrzędnego.

Począwszy od wersji 7.0 National Instruments dostarczał wsparcie dla instalacji systemu operacyjnego czasu rzeczywistego (PharLap) na klasycznym komputerze PC, a następnie możliwość programowania go z poziomu LV. Było to o tyle cenne, iż niektóre karty rozszerzeń występowały jedynie z magistralą PCI (która była niedostępna w komputerach czasu rzeczywistego PXI oferowanych przez NI) oraz dlatego, że karty PCI są tańsze od ich odpowiedników w standardzie PXI. Najważniejszym jednak powodem było to, że koszt systemu czasu rzeczywistego opartego o kontroler PXI był większy w porównaniu z jego odpowiednikiem opartym na komputerze PC. W lekkich warunkach przemysłowych nie zawsze istniała konieczność stosowania „pancernych" obudów PXI. Do komputera PC *PharLap* mógł zostać załadowany z dyskietek startowych (tworzonych w kolejnym narzędziu NI), co powodowało, iż komputer był widziany z poziomu LV jako kontroler czasu rzeczywistego. W teorii wyglądało to bardzo ładnie jednak praktyka pokazywała, że nie każdy sprzęt nadaje się do pracy pod kontrolą *PharLap*. Kończyło się to zwykle empirycznymi poszukiwaniami źródła problemu, a więc wymianą poszczególnych elementów komputera. W wersji 8.20 NI przygotował spore uproszczenie tej procedury w postaci tzw. *System Validatora*. Po załadowaniu do potencjalnego przyszłego komputera RT, program sprawdza wszystkie jego podzespoły pod kątem przydatności do pracy z systemem czasu rzeczywistego i wskazuje możliwe źródła przyszłych kłopotów. Okazuje się, że jednym z najczęstszych problemów uniemożliwiających takie użycie komputera PC jest karta sieciowa oraz brak wsparcia dla niej przez system operacyjny, dlatego National Instruments przygotował specjalny zestaw karty sieciowej i sterownika dedykowanego do pracy z modułem LV RT, nosi on nazwę *Real–Time Desktop Bundle*.

Dwie powyższe zmiany w LabVIEW 8.20 w stosunku do 8.0 mają charakter ewolucji powodującej ułatwienie obsługi już istniejących modułów LabVIEW. Zupełną nowością jest natomiast *LabVIEW Touch Panel Module* dostarczający wsparcie dla przemysłowych paneli dotykowych zarówno takich, w których kryje się kompletny kontroler (pracujący z Windows CE bądź XP), jak również dla dotykowych paneli będących jedynie interfejsami użytkownika (tzw. *Human Machine Interface* – HMI) na przykład dla kontrolerów PXI. LabVIEW *Touch Panel Module* stanowi dopełnienie znanego już z wersji 7.0 LabVIEW PDA Module przeznaczonego dla palmtopów.

Usprawnienia w pakiecie LabVIEW nie ominęły także *LV Embedded Development Module* (omówionego na łamach EP+ o ARM) pozwalającego programować w LV praktycznie każdy 32–bitowy procesor. Począwszy od wersji 8.20, stworzone przy jego pomocy platformy wykonywalne (*execution target*) LV są traktowane jak każda inna platforma sprzętowa NI, a więc mogą być zarządzane i wykorzystywane z poziomu *Project Explorera*. Jest to kolejny krok w stronę rozszerzalności rozproszonych systemów kontroli i akwizycji danych. Z okazji jubileuszu przygotowano też nowe przykłady portów LV na platformę ARM7 oraz procesory DSP Texas Instruments z rodziny C6000.

Oprogramowanie LV 8.20 wyposażono w oczekiwaną od dawna (spodziewano się jej już od wersji 8.0) możliwość *Programowania Orientowanego Obiektowo*. Funkcjonalność OOP (*Object–Oriented Programming*) ma znaczenie przy tworzeniu bardzo dużych aplikacji, gdzie wymagana jest modułowość oraz bezpieczeństwo i ochrona zmiennych należących do obiektów innych klas (czyli tzw. enkapsulacja). LV wspiera sztandarowe funkcjonalności OOP, a więc tworzenie klas i dziedziczenie. Jest to ukłon w stronę programistów tradycyjnych języków programowania, np. C++ czy Java, gdzie technologia programowania obiektowego jest jedynym właściwym

sposobem tworzenia aplikacji. NI zresztą zaznacza, że możliwość ta nie jest dla każdego, nie wydaje mi się też, aby LV samo w sobie stanowiło dobre środowisko do nauki tworzenia programów orientowanych obiektowo. Niemniej OOP jest bardzo interesującą opcją mającą na celu zmianę tendencji postrzegania LV nie jako środowiska typowo programistycznego. Następnym usprawnieniem samego procesu programowania w LV8.20 jest *DLL Import Wizard*. Kreator ten pozwala na automatyczne importowanie funkcji zawartych w bibliotekach *dll* (*Dynamic Linked Library*). Wymaga on podania jako informacji wejściowej ścieżki dostępu do pliku nagłówkowego *.h* biblioteki *dll*, a jako rezultat dostarcza gotowych prototypów funkcji i skonfigurowanych bloków *Call Library Function Node*. Dotychczas to na programiście spoczywało prawidłowe dobranie typów zmiennych przyjmowanych przez funkcję, jak również typów zmiennych zwracanych. Narzędzie to tworzy osobny plik .vi dla każdej funkcji zawartej w bibliotece, który następnie można bezpośrednio umieścić w tworzonym kodzie

DLL Import Wizard powinien zatem zminimalizować czas wdrożenia funkcji bibliotecznych poprzez wykorzystanie informacji zawartych w samych bibliotekach, jak i w ich plikach nagłówkowych. Warto dodać, że funkcjonalność stosowania *dll* w LV zaimplementowano już w wersji 4, a obecnie istnieje wsparcie tej technologii zarówno dla Windows jak i Mac OS oraz dla Linuxa. Podążając za najnowszymi trendami NI stworzyło też *.NET Web Service Import Wizard*, pozwalający w łatwy sposób importować procedury i funkcje wykonywane na zdalnych serwerach i używać ich w lokalnej aplikacji, taka funkcjonalność pozwala na wykorzystywanie zasobów sieci i nie ogranicza nas do używania funkcji standardowo dostarczanych z LV. Ostatnim już kreatorem LV 8.20 jest *Instrument Driver Export Wizard* pozwalający eksportować np. sterowniki urządzeń napisanych w LV do biblioteki *dll* oraz stworzenie pliku deklaracji funkcji *.h*. Należy zaznaczyć po raz kolejny, że funkcjonalność taka była dostępna w LV od dość dawna, jednak nie w tak przystępnej formie.

Podsumowanie

Jeśli przyjrzymy się z bliska wszystkim możliwościom LV możemy dojść do wniosku, że nie ma takiej platformy sprzętowej, na której nie mogłoby ono być użyte, nie ma takiej technologii software'owej, której LV nie wspierałoby i nie ma takiej grupy programistów, o którą LV nie zabiegałoby. Przy tak szerokim zakresie przeznaczenia i tak dynamicznie tworzonych rozszerzeniach do podstawowego pakietu LV, musiał powstać pewien bałagan. LabVIEW 8.20 wydaje się być próbą uporządkowania całego dorobku, jaki NI wypracował rozwijając kolejne moduły wspierające różne technologie. Z drugiej jednak strony wszystkie kreatory dostarczone z LV 8.20 i ułatwienia mają na celu spopularyzowanie programowania graficznego i przyciągnięcie nowych użytkowników, których na pierwszy rzut oka mogą odstraszać tak ogromne możliwości LV. Patrząc na LV 8.20 z tej strony, można śmiało stwierdzić, że nie jest ono rewolucją a jedynie ewolucją – w moim przekonaniu, niezwykle udaną.

Marcin Chruściel, EP
marcin.chrusciel@ep.com.pl