Pomiary oscyloskopowe: okiem praktyka, część 10

Dziesiątą część cyklu artykułu poświęcamy pokazaniu praktycznych zagadnień związanych z komunikacją nowoczesnych oscyloskopów firmy Tektronix z komputerami PC. Na płycie CD–EP10/2007B publikujemy oprogramowanie narzędziowe i przykładową aplikację opracowaną przez autora.

Komunikacja oscyloskopu z komputerem

Rezultaty pomiarów wykonanych oscyloskopem cyfrowym można archiwizować na kilka sposobów. Najprostszym z nich, i prawie zawsze dostępnym, jest zapis danych w pamięci referencyjnej. Jej pojemność jest jednak zwykle ograniczona do kilku przebiegów. Coraz częściej oscyloskop cyfrowy oferuje zapis na zewnętrznym nośniku, jak np. karty Compact Flash. Zapisuje się zarówno obraz z ekranu do pliku graficznego, dane z rekordu akwizycji, jak i komplet nastaw przyrządu. O wiele większą funkcjonalność oferuje połączenie oscyloskopu z komputerem. Do niedawna była to dodatkowa opcja, obecnie takie rozwiązanie staje się prawie standardem. W prostych modelach mamy do dyspozycji połączenie poprzez RS232, GPIB, LAN lub USB. Bardziej zaawansowane konstrukcje wyposażone są we własny komputer PC, gdzie komunikacja odbywa się np. poprzez wewnętrzny interfejs PCI. Wraz z oscyloskopem otrzymujemy napisany przez producenta program komunikacyjny. Jego możliwości nie zawsze zgodne są z naszymi potrzebami i oczekiwaniami. W tym odcinku postaram się przekonać Czytelników, że napisanie własnej aplikacji może być bardzo łatwe i szybkie.

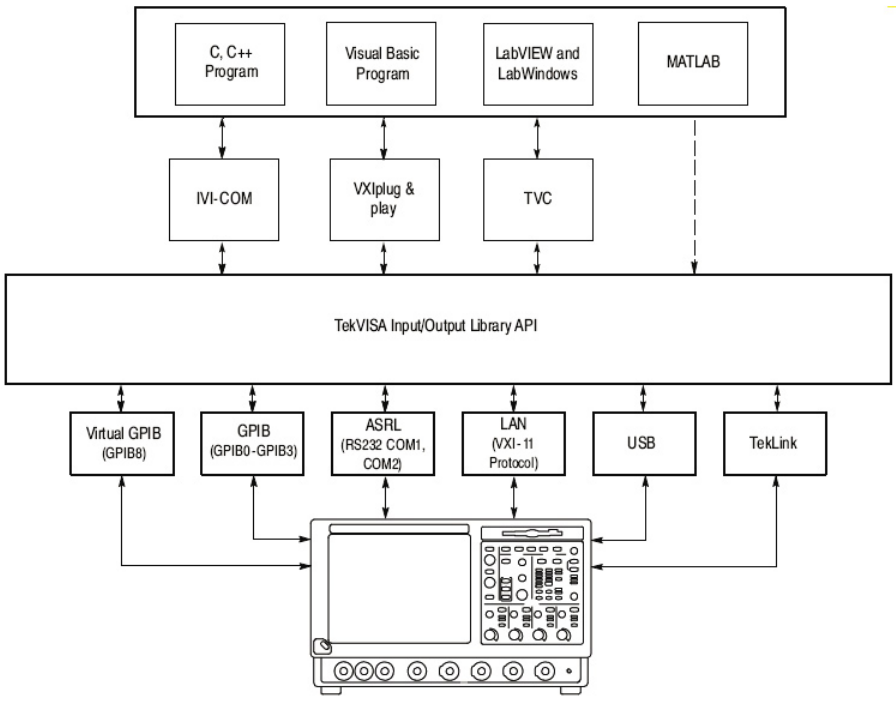
Przedstawiony w dalszej części przykład dotyczy oscyloskopów Tektronix TDS1000B i TDS2000B wyposażonych fabrycznie w port USB. Nic jednak nie stoi na przeszkodzie, aby podobne rozwiązanie zastosować do wcześniejszych modeli tego producenta. Warunkiem jest jednak jakikolwiek port komunikacyjny, który w poprzednich wersjach oscyloskopów dostępny był jako dodatkowa opcja. Potrzebny także będzie komputer PC z systemem Windows.

Napisanie własnej aplikacji komunikacyjnej bazuje na wykorzystaniu funkcji biblioteki TekVISA, dostarczanej wraz z nowym przyrządem. Można ją także bezpłatnie pobrać z internetowej strony producenta (*www.tek.com*). Wersja 3.02 jest dostępna na płycie CD–EP120/2007B. Plik instalacyjny ma rozmiar 75 MB (w najnowszej wersji 3.3 nawet 109 MB). Zawiera on jednak dodatkowe składniki, m.in. kilka aplikacji oraz wiele przykładów w języku C. Po zainstalowaniu znajdują się one w domyślnym katalogu *C:\VXIpnp*.

TekVISA jest opracowaną przez Tektronix implementacją biblioteki I/O VISA (*Virtual Instrument Software Architecture*). Ogólny schemat jej udziału w komunikacji pomiędzy przyrządem pomiarowym a środowiskiem, w którym tworzy się własną aplikację, przedstawiono na **rys. 53**. Jak widać, program użytkownika odwołuje się po prostu do interfejsu programowania TekVISY. Dokładnie jest on opisany w instrukcji obsługi „*TekVISA Programmer Manual*".

Wbrew pozorom, napisanie własnego programu komunikacyjnego jest dość proste. Ponieważ przykłady w C oraz Visual Basicu dostarczone są przez producenta, w artykule przedstawiono wykorzystanie środowiska Delphi. Bezpłatne wersje Delphi Personal są dostępne na stronie firmy Borland. Autor korzystał ze starszej wersji Delphi 5.0 Standard.

Przed przystąpieniem do tworzenia programu należy zainstalować z poziomu Delphi odpowiednią kontrolkę ActiveX. Zakładając, że TekVISA jest już zainstalowana wcześniej, wykonuje się to poprzez wybranie z menu *Component–>Import ActiveX Control...*. W oknie dialogowym należy wybrać



Rys. 53.

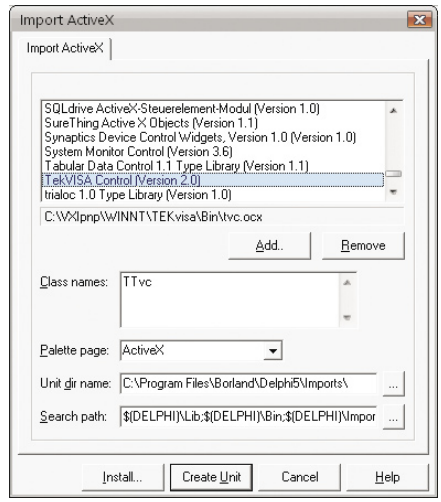
TekVISA Control (przedstawiono to na **rys. 54**) oraz wcisnąć *Install…* i potwierdzić w pojawiającym się oknie *Install*. Od tego momentu w zakładce ActiveX palety komponentów dostępny będzie komponent Tvc (*TekVISA Control*). Opis związanych z nim właściwości, metod oraz zdarzeń znajduje się pliku pomocy *Tvc.chm* w katalogu *C:\VXIpnp\WINNT\TekVISA\Bin*.

Po zainstalowaniu TekVISY warto uruchomić aplikację *InstrumentManagera*, wybrać *SearchCriteria* i pozostawić tylko te interfejsy, które będą wykorzystywane. Instrument Manager uruchamia się poprzez menu kontekstowe ikony VISA w podajniku systemowym lub poprzez *Start–>Programy–>TekVISA*.

Mając już przygotowane środowisko pracy, przystąpmy do napisania prostego programu, umożliwiającego odbieranie obrazów z ekranu oscyloskopu. Na początek, na czystej formie nowego projektu umieszczamy kolejno cztery komponenty: kontrolkę *Tvc*, przycisk polecenia *Button* oraz dwie listy *ListBox1* i *ListBox2*. Po wciśnięciu przycisku lista pierwsza powinna zawierać spis podłączonych przyrządów, a lista druga ich identyfikację. W tym celu w obsłudze zdarzenia *OnClick* przycisku wpisujemy kod widoczny na **list. 1**.

List. 1.

```
var
  i: integer;
  dev: Variant;
begin
  Tvc.SearchCriterion:= 0;
  dev:= Tvc.FindList;
  ListBox1.Clear;
  ListBox2.Clear;

  // dev zawiera teraz listę podłączonych urządzeń,
  // sprawdzamy czy jakiekolwiek są aktualnie dołączone

  if (varType(dev) <> varEmpty) then
  begin

  // identyfikujemy każde z połączeń oraz urządzeń dołączonych

    for i:= 1 to VarArrayHighBound(dev, VarArrayDimCount(dev)) do
    begin
      ListBox1.Items.Add(dev[i]);
      Tvc.Descriptor:= dev[i];
      ListBox2.Items.Add(Tvc.Query('*IDN?'));
    end;

  // jeśli brak jest połączeń wyświetlamy informację o tym

  end else ListBox1.Items.Add('Brak połączenia z oscyloskopem');
end;
```

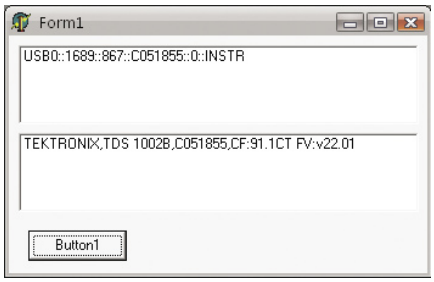


Rys. 54.

FindList umieszcza listę wszystkich podłączonych urządzeń w tablicy wariantowej *dev*. Jeśli lista ta okaże się pusta, informujemy o tym poprzez wyświetlenie odpowiedniego komunikatu w polu listy *ListBox1*. W przeciwnym wypadku wpisujemy tam spis wszystkich połączeń. Do pola *Tvc.Descriptor* wpisujemy nazwę kolejnego połączenia i od tej pory *Tvc* odnosi się do urządzenia związanego z tym połączeniem. Za pomocą funkcji *Query* wysyłamy do każdego z urządzeń zapytanie **IDN?* i odczytujemy identyfikujący je ciąg znaków. Zapisujemy go następnie na odpowiedniej pozycji listy *ListBox2*. Jeśli do komputera podłączony jest jeden oscyloskop, to wynik działania programu powinien być podobny do przedstawionego na **rys. 55**.

Aby skomunikować się z dowolnym dołączonym do komputera przyrządem, należy do pola *Tvc.Descriptor* wpisać nazwę danego połączenia dostępną poprzez *dev[i]*. Następnie, za pomocą *Tvc.WriteString* zapisać wymagane polecenie i korzystając z *Tvc.ReadString* odczytać ewentualną odpowiedź. Spis komend dla oscyloskopów Tektronix, TDS200, TPS2000 i TDS1000/TDS2000 znajduje się w podręczniku programowania dostępnym na internetowej stronie producenta oraz na płycie dołączonej do tego wydania EP.

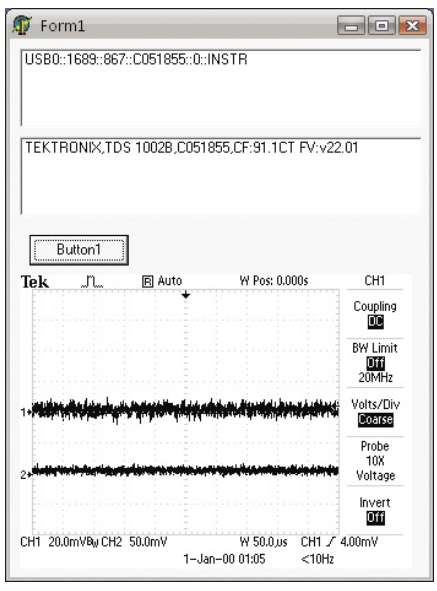
Pobranie obrazu z ekranu oscyloskopu do schowka systemu Windows wydaje się banalne. Obiekt *Tvc* posiada bowiem metodę *HardcopyToClipboard*. Odczytany rysunek wystarczy tylko przenieść do np. obiektu klasy *TImage*. Należy pamiętać, że aby odwoływać się do systemowego schowka, trzeba w sekcji *uses* dodać



Rys. 55.

ClipBrd. Jak się okazuje w praktyce, nie jest to najlepsza droga. Co prawda obraz można odczytać i przenieść w dowolne miejsce, jednak czasem jest to ostatnia, dająca się wykonać operacja. Powtórne wywołanie *HardcopyToClipboard* zazwyczaj skutkuje zawieszeniem programu lub wygenerowaniem błędu I/O. Co więcej, dostarczany wraz z TekVISĄ pasek narzędzi *WordToolbar*, zachowuje się w taki właśnie sposób podczas kopiowania obrazów z ekranu. W celu prześledzenia tego, co tak naprawdę robi wspomniana procedura można wykorzystać program *CallMonitor*. Uruchamia się go z menu kontekstowego ikony VISA, widocznej w podajniku systemowym. Problem jednak w tym, że przy uruchomionym *CallMonitorze* wszystko przebiega poprawnie.

Na szczęście istnieją też inne sposoby pobrania obrazu. Wykorzystamy pobieranie do pliku, a następnie wyświetlenie jego zawartości za pomocą metody *LoadFromFile*. Nie jest to szczególnie wyrafinowane rozwiązanie, niemniej skuteczne. Aby obraz z ekranu oscyloskopu przenieść do obiektu *TImage* wystarczy zaledwie kilka linii kodu widocznych na **list. 2**.



Rys. 56.

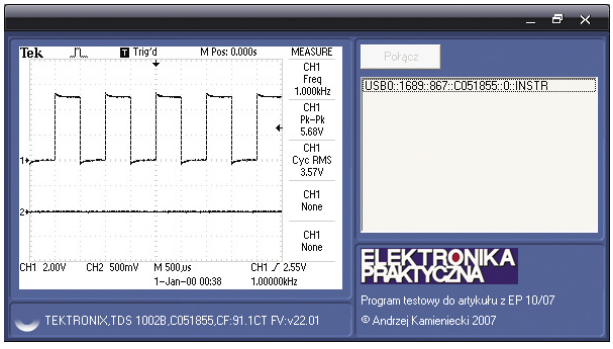
List. 2.

```
var
  retCnt: longint;
begin
  Tvc.WriteString('Hardcopy Start');
  Tvc.ReadToFile('temp', 80000, retCnt);
  Image.Picture.Bitmap.LoadFromFile('temp');
end;
```

Wysłanie komendy *Hardcopy Start* spowoduje skopiowanie aktualnej zawartości ekranu. Metoda *ReadToFile* pozwala naszemu programowi odczytać dane do pliku o nazwie *temp*, który domyślnie zostanie założony w tej samej lokalizacji, co uruchomiony program. Drugi parametr funkcji *ReadToFile* określa maksymalny rozmiar pliku w bajtach. Mapa bitowa pobrana z oscyloskopu TDS1000 ma rozmiar ok. 77 kB, stąd widoczne 80000. Jeśli teraz do projektu dodamy obiekt *TImage* oraz dopiszemy kod z list. 2, to przy dołączonym do komputera oscyloskopie (przy tylko jednym przyrządzie *Tvc* odnosi się właśnie do niego) otrzymamy rezultat podobny do przedstawionego na **rys. 56**. Kolejne wciskanie przycisku spowoduje pobranie następnego obrazu. W zależności od interfejsu komunikacyjnego, może to trwać kilka lub nawet kilkadziesiąt sekund. Dla połączenia poprzez USB jest to czas rzędu 3…5 s. Niestety, komunikacja tego typu poprzez TekVISĘ nie pozwoli na pracę całkowicie *on–line*.

Przedstawione rozwiązanie ma jednak poważną wadę. Podczas pobierania danych do pliku, aplikacja nie reaguje na polecenia użytkownika. Gdyby kolejny obraz miał być pobierany zaraz po poprzednim, to w takich warunkach ciężko byłoby nawet zamknąć program. Wstawienie *Application.ProcessMessages* nie na wiele się przydaje. Istnieje jednak i na to rada. Pobieranie danych do pliku należy zrealizować w osobnym wątku. W tym celu z menu *File–>New* trzeba wybrać *ThreadObject*. Kod z list. 2. umieszczamy w procedurze *Execute* nowego wątku, pamiętając o wpisaniu nazw modułów w odpowiednich sekcjach *uses*. Pewnej ostrożności wymaga zakończenie wątku. Aby program poprawnie zamknąć w dowolnym momencie, należy najpierw dokończyć trwający proces pobierania danych, zakończyć wątek i dopiero zamknąć program. Na płycie CD–EP10/2007B znajduje się przykładowy projekt przygotowany przez autora. Po wciśnięciu przycisku *Połącz* pobiera on nazwy wszystkich aktualnych połączeń. Następnie dla pierwszego przyrządu z listy (zakładamy, że jest to oscyloskop wymienionej na wstępie serii) wykonuje odczytywanie obrazu z ekranu i wyświetla go. Od tego momentu do chwili zamknięcia programu pobieranie jest cykliczne. Podczas pobierania danych aplikacja zachowuje 'mobilność' umożliwiając np. zmianę rozmiarów okna. Wynik jej działania przedstawia **rys. 57**. Na etapie przedstawionym w artykule, program komunikacyjny nie wykonuje nic szczególnego, ale może stanowić podstawę dla bardziej zaawansowanych zadań. Poza obrazami z ekranu, TekVISA umożliwia pobieranie danych z rekordu akwizycji. Stąd już tylko krok do generowania dowolnych przebiegów matematycznych i napisania aplikacji do np. pomiaru mocy strat w elemencie przełączającym. W poprzednim odcinku przedstawiony był wynik działania takiego programu, stworzonego w firmie Tektronix. Mam jednak nadzieję, że zaprezentowany prosty przykład zachęci Czytelników do napisania własnej wersji, tym bardziej, że środowisko Delphi oferuje w tym zakresie chyba największe ułatwienia. Uzyskanie funkcjonalności podobnej do aplikacji pomiarów mocy dla oscyloskopów serii TPS2000, można uzyskać poświęcając jeden z nadchodzących, długich zimowych wieczorów.

Andrzej Kamieniecki



Rys. 57.