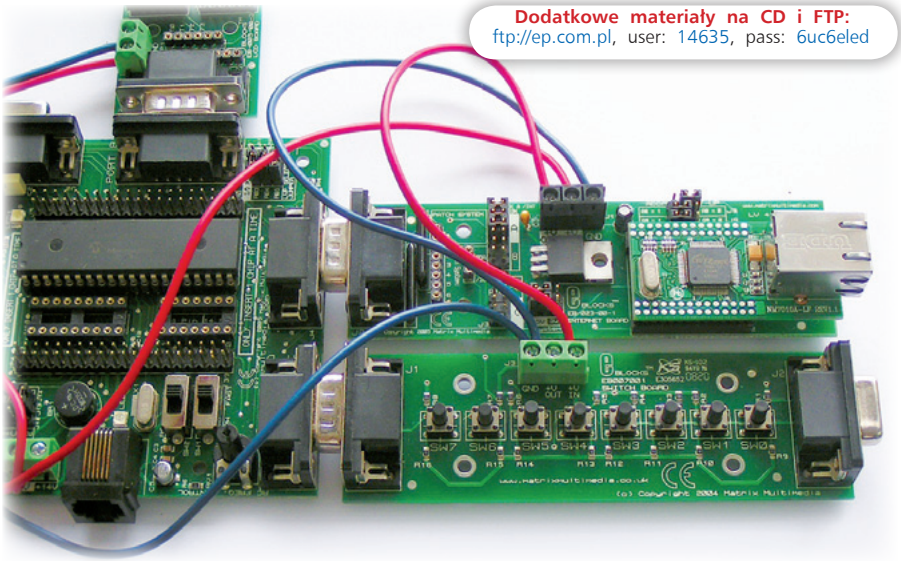
Sieciowe możliwości Flowcode

Aplikacje sieciowe dla mikrokontrolerów to aktualnie jeden z najgorętszych tematów. Mimo, że rozwiązania stosowane w sieciach komputerowych LAN nie należą do prostych, to producenci mikrokontrolerów i specjalizowanych układów peryferyjnych robią wszystko, żeby połączenie mikrokontrolera do sieci było tak proste, jak to tylko możliwe.



Fotografia 1. Płyta główna E-block dla mikrokontrolerów PIC z podłączonymi modułami

Niektóre firmy udostępniają gotowe biblioteki zawierające stosy TCP/IP. Tak robi na przykład Microchip. Jej rozwiązanie ma szereg trudnych do przecenienia zalet: zaimplementowano większość protokółów z rodziny TCP/IP, a programista może użyć tylko tych, które są mu potrzebne. Jednak (i ta uwaga dotyczy nie tylko Microchipa) pomimo dostępnych, dobrze udokumentowanych przykładów i materiałów szkoleniowych, korzystanie z programowego stosu wymaga od programisty pewnego przygotowania i doświadczenia. Początkującemu raczej będzie trudno skorzystać z tego rozwiązania.

Dobrą alternatywą są układy peryferyjne z wbudowanym sprzętowym stosem TCP/IP. Sprzętowy interfejs układu dba o obsługę warstw fizycznej, sieciowej i transportowej, pozwalając programiście skupić się tylko na warstwie aplikacji. Jednym z takich układów jest W3100A firmy WIZnet. Oprócz wbudowanych protokółów internetowych TCP/IP, UDP, ICMP i ARP, układ wspiera protokóły DLC i MAC interfejsu Ethernet, a komunikuje się z nadrzędnym mikrokontrolerem przez interfejs I²C lub magistralę równoległą. W3100A został zastosowany w jednym z modułów systemu E– block oferowanego przez firmę Matrix Multimedia producenta pakietu Flowcode.

Co to jest E-block?

System E-block dla mikrokontrolerów PIC składa się płyty głównej, na której umieszczono obwody zasilania, mikrokontroler i programator z interfejsem USB (**fotografia 1**). Linie portów dołączono do żeńskich, 9-wyprowadzeniowych złączy D-SUB. Przez nie do płyty głównej można dołączyć układy peryferyjne systemu E-block: wyświetlacz LCD, przyciski, diody LED, przekaźniki, gniazdo kart SD, układy Bluetooth, RFID i wiele innych.

Wspomniany moduł z układem W3100A jest również do płyty głównej dołączany złączem D-SUB9 (**fotografia 2**). Do połączenia modułu z siecią Ethernet służy standardowe gniazdo RJ-45.

Oprócz minimodułu z układem W3100A, na płytce umieszczono stabilizator napięcia 3,3 V, złącza zasilania, zwory łączące sygnały sterujące z liniami portów i do ustawiania adresu *slave* interfejsu I²C.

W3100A zawiera kompletny stos protokółów TCP/IP, ale jego programowa obsługa nie jest banalna. Jak łatwo domyślić się, Flowcode zdejmuje z programisty ciężar wnikliwego wczytywania się w dokumentację, poznawania wszystkich rejestrów sterujących i algorytmów ich programowania.

Ponieważ dysponuję płytą główną E-block dla mikrokontrolerów PIC i kilkoma modułami peryferyjnymi w tym internetowym modułem EB-023, postanowiłem sprawdzić, czy można w miarę prosto i szybko dołączyć mikrokontroler PIC do sieci. Jako pierwszy wybrałem mikrokontroler PIC16F877A i dodatkowo moduły: wyświetlacza LCD EB-005 (dołączyłem go do portu B) i styków EB-007 (dołączyłem go do portu D). Moduł internetowy korzysta z interfejsu I²C i dlatego został połączony z portem C.

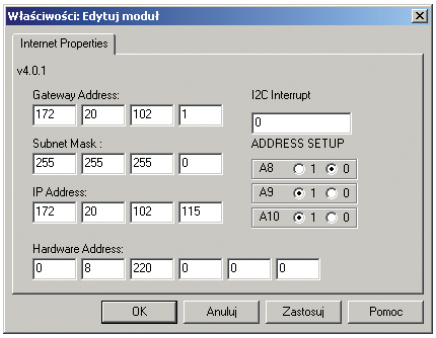
Obsługa modułu internetowego jest powiązana z komponentem *TCP_IP* dostępnym w najnowszej wersji *Flowcode V4* dla mikrokontrolerów PIC. Można go wybrać poprzez zakładkę *Peripherial*. Komponent *TCP_IP* zawiera wiele makr sprzętowych i aby móc go efektywnie wykorzystać trzeba mieć trochę wiedzy na temat protokółów internetowych.

PING

Polecenie PING jest używane do diagnozowania połączenia pomiędzy dwoma stacjami. Do jego przesłania służy protokół ICMP. Ramka ICMP składa się 8-bitowych pól: type, code, 16-bitowej sumy kontrolnej i opcjonalnego pola danych. Polecenie PING to inaczej żądanie echa. Stacja wysyłająca to żądanie wysyła pakiet ICMP z typem 0x08 (*Echo Request*). Stacja pytana odsyła potwierdzenie z typem 0x00 (*Echo Reply*). Kod ma wartość 0x00, a pole danych zawiera 2-bajtowy identyfikator, 2-bajtowy numer sekwencyjny kolejnych pakietów oraz dowolne inne dane. Stacja, która odebrała żądanie echa, odsyła w polu danych te same informacje, które odebrała. Identyfikator i numer sekwencyjny mogą służyć do identyfikacji czy odebrane potwierdzenie *Echo Reply* jest odpowiedzią na konkretne żądanie.

Ramka protokółu ICMP jest umieszczana w polu danych protokołu IP. W ramce protokółu IP będzie nas interesowało pole *Protocol*, czyli typ protokołu, którym przesyłane są dane w polu danych. Dla protokołu ICMP pole *Protocol* ma wartość 0x01, a dla protokółu TCP ma wartość 0x06.

Fotografia 2. Moduł internetowy E-block



Rysunek 3. Okno konfigurowania komponentu TCP_IP

Rysowanie programu

Rysowanie programu zaczynamy od umieszczenia na panelu elementu *TCP_IP* i kliknięcia na nim prawym klawiszem. Z rozwijanego menu wybieramy *Ext Properties*. W oknie *Properties* (**rysunek 3**) musimy wpisać wartości nastaw bramy domyślnej, maski podsieci i adresu IP modułu. Dodatkowo ustawia się tutaj adres MAC oraz wartość *Address Setup*. Jest to adres slave układu W3100A w magistrali I²C. Jego wartość musi być zgodna z ustawieniami zworek J8, J9 i J10 na module EB-023. Pingowanie możemy przeprowadzić łącząc moduł np. bezpośrednio z komputerem PC. Można też moduł wpiąć do routera lub switch'a sieciowego. Obie te metody są odpowiednie dla celów testowych, ale trzeba zwrócić uwagę na kabel połączeniowy, ponieważ połączenie modułu z komputerem PC wymaga, aby kabel był z przeplotem.

Uwaga: moduł nie ma zaimplementowanej funkcji klienta DHCP i adres trzeba wpisać ręcznie. Po wpisaniu danych w oknie *Properties* klikamy na Ok i zamykamy okno. W tym momencie można używać makr komponentu *TCP_IP* w programie.

Na początku programu trzeba wywołać makro *Initialize*. Inicjalizuje ono układ W3100A i ustawia wcześniej wpisane w oknie *Properties* parametry.

Przesyłanie danych musi być poprzedzone zainicjowaniem gniazd transmisji. Można zdefiniować do 4 takich gniazd (*sockets*). My będziemy przesyłać ramki ICMP z użyciem protokołu IP i dlatego musimy zdefiniować gniazdo (*socket*) IP. Do tego celu jest używana funkcja *Create IP_Socket* z parametrami: numer gniazda, protokół, adres rozgłaszeniowy (*brodcast*).

Numer gniazda może mieć wartości od 0 do 3. W polu protokół wpisujemy 0x01, bo taką wartość będzie miało pole *Protocol* w nagłówku ramki protokołu IP dla ICMP. Rozgłaszanie włącza się wpisując w parametr 0x01, a wyłącza wpisując 0x00. Na koniec pozostaje zdefiniowanie adresu IP stacji, od której będziemy żądać przesłania echa. Adres IP stacji docelowej ustawia się funkcją *Set_Destination*. Jej argumentami są adres IP i 2 bajty określające numer portu.

Po wywołaniu powyższych funkcji, program jest gotowy do przesyłania danych w protokole ICMP. Wysyłanie danych musi być poprzedzone wywołaniem *Tx_start* z argumentem określającym numer kanału, w którym będą transmitowane dane. Potem można już wysyłać kolejne bajty ramki ICMP. Pamiętamy, że stacja, która inicjuje żądanie

Listing 1. Kod źródłowy strony głównej serwera

```
<HTML>
<HEAD>
  <TITLE>Main page</TITLE>
</HEAD>
<BODY>

<P><CENTER><B>Test Webserver Flowcode PIC Micro</B></CENTER></P>

<P><CENTER><TABLE WIDTH="450" BORDER="1" CELLSPACING="2" CELLPADDING="2">
  <TR>
    <TD WIDTH="50%">
    <A HREF="http://172.20.102.115/">Index</A></TD>
    <TD WIDTH="50%">    Home</TD>
  </TR>
  <TR>
    <TD WIDTH="50%">
    <A HREF="http://172.20.102.115/page2">Page 2</A></TD>
    <TD WIDTH="50%">    Wyswietl wartosc zmiennej .</TD>
  </TR>
  <TR>
    <TD WIDTH="50%">
    <A HREF="http://172.20.102.115/page3">Page 3</A></TD>
    <TD WIDTH="50%">    Test strony 3.</TD>
  </TR>
  <TR>
    <TD WIDTH="50%">
    <A HREF="http://172.20.102.115/page4">Page 4</A></TD>
    <TD WIDTH="50%">    Test strony 4.</TD>
  </TR>
</TABLE></CENTER></P>

</BODY>
</HTML>
```

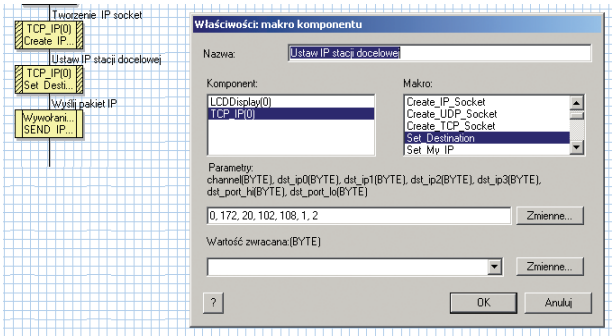
Listing 2. Kod w html strony page 2

```
<!DOCTYPE HTML PUBLIC „-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>Internet E-Block Component</title>
</head>
<P><CENTER><TABLE WIDTH="450" BORDER="2" CELLSPACING="4" CELLPADDING="2">
<body BGCOLOR="#FF3333">
<font face="arial">
 <TR BGCOLOR="#3333FF">
<TD WIDTH="50%">
  <H1>Wartosc zmiennej  = %VALUE%</H1>
</TD WIDTH="50%">
  </TR>
<TR BGCOLOR="#6666FF">
    <TD WIDTH="50%">
    <A HREF="http://172.20.102.115/">Index</A></TD>
    </TD WIDTH="50%">
  </TR>
</font>
</body>
</TABLE></CENTER></P>
</html>
```

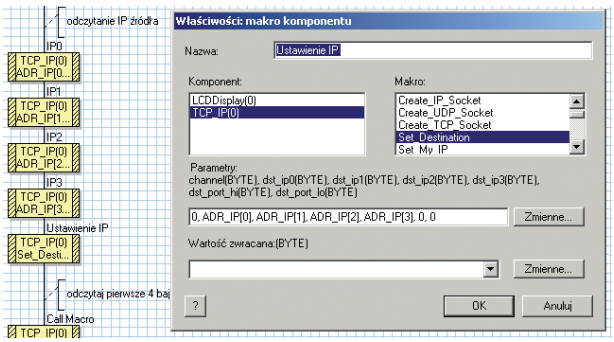
echa (PING) wysyła ramkę z typem *Echo Request* (0x08). Program przesyła bajty wywołując funkcję *Tx_sendbyte* z argumentami określającymi numer kanału i przesyłany bajt.

Jeżeli w konfiguracji nie popełniliśmy błędów (adresy, maska, brama), to pakiet powinien zostać odebrany przez stację o adresie ustalonym w funkcji *Set_destination*. Stacja odeśle do naszego modułu potwierdzenie z typem *Echo Reply* (0x00). Nasz moduł musi teraz to potwierdzenie odebrać, żeby stwierdzić czy realizacja polecenia PING przebiegła prawidłowo.

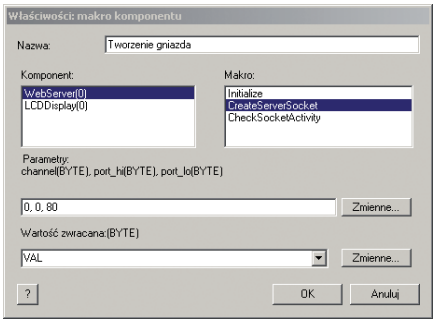
Procedura odbioru musi również odbierać żądanie echa wysłane przez inna stację. Jeżeli pierwszy odebrany bajt ma wartość 0x08, to oznacza że moduł odbiera żądanie echa *Echo Request* wysłane przez inną stację (komputer). Jeżeli odebrany bajt jest 0, to zaczynamy odbierać potwierdzenie *Echo Reply* na nasze żądanie. Ponieważ wiemy jaki wysłaliśmy identyfikator i numer sekwencji, to procedura odbioru może te wartości testować. Można też sprawdzać prawidłowość sumy CRC pakietu.



Rysunek 4. Argumenty funkcji *Set Destination*



Rysunek 5. Ustawienie adresu IP odczytanego z nagłówka



Rysunek 6. Okno właściwości komponentu Webserver

W czasie testu użyłem procedury, która odbiera cała ramkę, ale nie testuje ani identyfikatorów, ani sumy CRC, tylko wyświetla osiem pierwszych znaków ASCII z pola danych.

Trochę bardziej skomplikowany jest fragment odbierania *Echo Request* i wysyłania potwierdzenia do nadawcy żądania. Moduł odbiera pakiet ICMP i umieszcza go w buforze odbiorczym. Żaby wysłać potwierdzenie *Echo Request* trzeba prawidłowo wyliczyć 2 bajty sumy kontrolnej CRC, a do tego potrzebna jest ilość danych pakietu ICMP. Liczbę bajtów w pakiecie można odczytać funkcją *Rx_readheader*. W trybie IP funkcja Rx_readheader z identyfikatorem idx=1 odczytuje liczbę bajtów zawartą w polu danych (w naszym przypadku liczba bajtów wchodzących w skład całego pakietu ICMP) i adres IP źródła pakietu IP.

CRC jest wyliczane z odebranych danych pakietu z tym ,ze pierwsze cztery bajty są wyzerowane, bo typ jest bajtem zerowym (*Echo Reply*), kod jest bajtem zerowym i nie liczymy CRC z dwu bajtów odebranego CRC. Liczeniem sumy kontrolnej zajmuje się makro programowe CRC.

Drugim ważnym elementem jest odczytanie adresu IP stacji, która wysłała *Echo Request*, po to, aby móc do niej odesłać *Echo Reply*. Ten adres jest zapisany w nagłówku pakietu IP jako *Destination Address*. Jak już wspomniałem, do jego odczytania wykorzystuje się funkcję *Rx_readheader* z identyfikatorami idx=2...5. Na **rysunku 4** pokazano fragment programu odczytujący IP z pakietu *Echo Request*, wysłanego do nas. Ten adres jest zapamiętywany w tablicy *ADR_IP*, a potem funkcją *Set Detination* wpisywany w pole *Destination Address* pakietu IP, który zostanie wysłany z pakietem ICMP w polu danych – rysunek 4.

Po obliczeniu sumy kontrolnej i ustawieniu adresu źródła można skompletować pakiet ICMP. W poleceniu PING w odsyłanym *Echo Reply* pole danych powinno mieć taką samą zawartość, jak w odebranym *Echo Request*. Po doczytaniu, cały pakiet ICMP jest zapamiętywany w buforze odbiorczym, który jest sekwencyjnie czytany funkcją *Rx_readbyte*. Wykorzystamy to do odesłania takich samych danych, jakie odebraliśmy. Żeby ustawić się na odebrane dane odczytujemy i dodajemy pierwsze cztery bajty, żeby licznik adresowy bufora odbiorczego ustawił się na pierwszy bajt odebranego pola danych. Potem wysyłamy dwa bajty zerowe, dwa bajty wcześniej wyliczonej sumy kontrolnej i wszystkie bajty pola danych.

Po wysłaniu wszystkich danych trzeba uruchomić funkcję *Rx_flush_data*. Po jej wykonaniu możliwe jest odebranie następnego pakietu.

Web Server

Jako kolejną aplikację przygotowano WebServer przeznaczony do zbudowania własnego serwera stron www. Komponent jest tak zbudowany, że nawet niezbyt zaawansowani użytkownicy są w stanie stworzyć własny serwer stron.

Do komponentu WebSerwer przypisano trzy makra sprzętowe: *Initalize*, *CreateSeverSocket* i *CheckSocketActivity*.

Makro *Initialize* zeruje i inicjalizuje moduł internetowy EB-023 oraz ustawia parametry sieciowe: bramę domyślną, maskę podsieci, adres IP i adres MAC. Ustawia się tu też maksymalną liczbę stron (z zakresu 1...4) i adres interfejsu I^2C układu W3100A.

Makro *CreateServerSocket* tworzy gniazdo do wysyłania i odbierania stron internetowych. Argumentami makra są numer kanału i numer portu. Dla gniazda można zdefiniować do czterech kanałów, które mogą być jednocześnie otwierane. Numer portu jest 16-bitowy (zakres 0...65535). Do przesyłania stron www w protokole HTTP jest zarezerwowany port 80.

Strona umieszczona na serwerze wymaga stałego monitorowania żądania dostępu na przykład dla stwierdzenia czy przeglądarka nie chce odświeżenia strony. Do tego celu służy makro *CheckSocketActivity*. Argumentem wejściowym jest numer kanału. Zależnie od skonfigurowania połączenia sieciowego przesłanie strony może wymagać jednego lub więcej żądań dostępu.

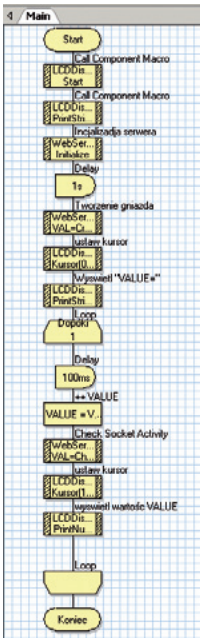
W zakładkach Web Page 1...4 trzeba umieścić kod strony napisany w html. Napisanie prostej strony bez grafiki nie jest trudne. Do prób z serwerem użyłem kodu źródłowego zamieszczonego na **listingu 1**. Program serwera jest zaskakująco prosty. Oprócz modułu internetowego użyłem modułu wyświetlacza LCD do częściowej kontroli poprawności przesyłania danych w sieci.

Na początku programu jest inicjalizowany wyświetlacz LCD i wyświetlany napis „Webserwer Test". Następnie jest wywoływane makro *Initalize* komponentu Web Serwer. Ponieważ inicjalizacja trochę trwa, to program odlicza opóźnienie wynoszące 1 sekundę. W kolejnym kroku przez wywołanie makra *CreateServerSocket* z argumentami 0 (numer kanału) i 80 (numer portu) jest tworzone gniazdo do komunikacji z portem 80 (**rysunek 6**).

W tym momencie możemy testować, czy klient (przeglądarka) nie żąda dostępu do strony na serwerze. Żeby test nie był testowaniem „niczego", to w podstronę *page2* został wbudowany mechanizm przesyłania do klienta wartości 8-bitowej zmiennej inkrementowanej co około 100 ms. Na listingu 2 zamieszczono kod podstrony *page2*.

Aby dana mogła być przesyłana z serwera do przeglądarki, to trzeba najpierw zdefiniować w programie nazwę zmiennej – w naszym przypadku jest to 8-bitowa zmienna o nazwie VALUE. Wartość tej zmiennej jest przesyłana z użyciem protokołu http do klienta (przeglądarki), po umieszczeniu jej nazwy w otoczeniu znaków %. U nas będzie to linijka *<H1>Wartosc zmiennej = %VALUE%</H1>*.

Tyle wystarczy, aby przeglądarka otrzymała instrukcję „wyświetl wartość zmiennej VALUE". Ten bardzo prosty mechanizm (prosty z punktu widzenia użytkownika) pozwala na odczytywanie z serwera wartości dowolnych zmiennych zdeklarowanych w programie po każdym poleceniu odświeżenia strony w przeglądarce.



Rysunek 7. Program Webserwera

Na **rysunku 7** pokazano kompletny program w postaci graficznej utworzony we Flowcode V4 dla mikrokontrolerów PIC Microchip. Wartość inkrementowanej zmiennej w pętli nieskończonej jest wyświetlana na wyświetlaczu LCD, tak aby można było sprawdzić, czy program wyświetlania na stronie działa prawidłowo.

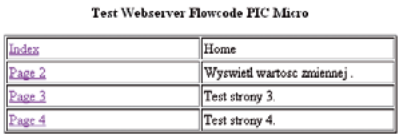
Do testowania działania Webserwera użyto mikrokontrolera PC18F458. Po zaprogramowaniu trzeba połączyć moduł z siecią Ethernet i w przeglądarce wpisać adres IP 172.20.102.115, ustalony dla modułu w trakcie inicjalizacji. Jeżeli wszystko zostało zrobione prawidłowo, to powinno się wyświetlić okno strony głównej, takie jak pokazano na **rysunku 8**.

Z poziomu strony głównej można przejść do 3 stron oznaczonych jako *Page2*, *Page3* i *Page4*. Wartość *VALUE* z modułu serwera jest odczytywana na stronie 2, otwierającej się po kliknięciu na *Page2*. Do strony głównej można wrócić klikając na *Index*.

Uwagi końcowe

Dzięki pracy wykonanej przez twórców pakietu Flowcode i układu WM3100A aplikacje sieciowe są tak proste, że może się próbować z nimi mierzyć nawet mało zaawansowany programista. Dowodzą tego pokazane tutaj dwa przykłady. Pierwszy z nich (ping) wymaga trochę szerszej wiedzy na temat protokołów sieciowych. Z kolei projekt Webserwera jest tak pomyślany, że właściwie potrzebne są tylko podstawy języka HTML i znajomość pakietu Flowcode.

Tomasz Jabłoński, EP
tomasz.jablonski@ep.com.pl



Rysunek 8. Fragment okna głównego strony Webserwera