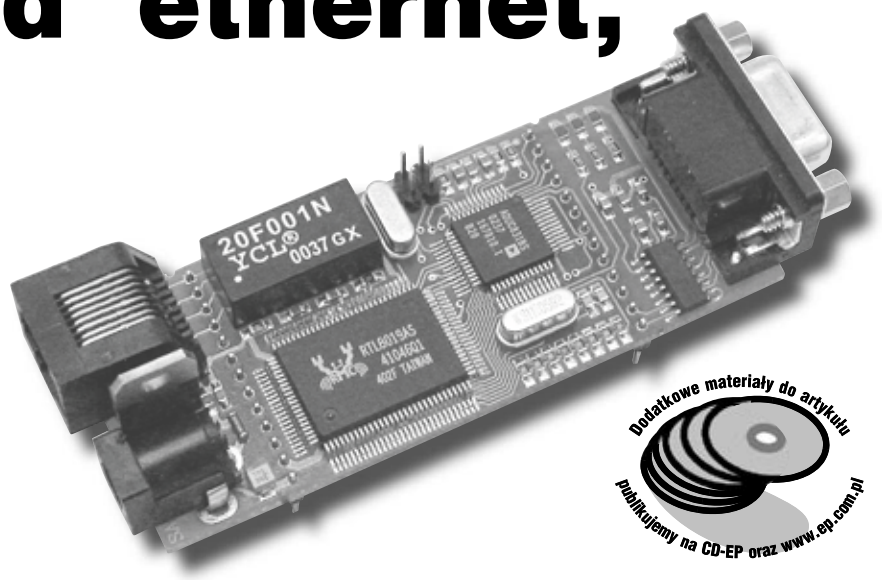
Embedded ethernet, część 4

AVT-547



Ostatnia część cyklu zawiera opis uruchomienia i przetestowania modułu sieciowego. Przedstawiono także możliwości jego rozbudowy i kierunki modyfikacji oprogramowania, które mogą znacząco rozszerzyć jego funkcjonalność.

Rekomendacje:

Artykuł polecamy wszystkim zainteresownym łącznością poprzez ethernet, którzy chcieliby zapoznać się z podstawami działania sieci i zdobycie tej wiedzy uwieńczyć samodzielnym wykonaniem mini-serwera sieciowego

Na początek

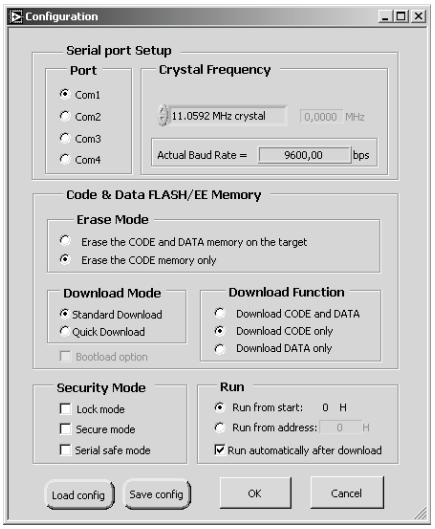
Zanim podłączymy zasilanie do naszego modułu sieciowego musimy przygotować kilka niezbędnych składników, które są konieczne do uruchomienia modułu. Na początek zasilanie. Jeśli wykonaliśmy płytkę z obsadzonym stabilizatorem, to potrzebujemy zewnętrznego zasilacza, dającego stałe napięcie w zakresie 7...12 V. Cały układ nie pobiera więcej niż 80 mA prądu. Jeśli zasilamy układ wprost z 5V (bez stabilizatora U1), to musimy zadbać, aby było ono pozbawione tętnień, niekorzystnie wpływających na parametry części analogowej.

Kolejną rzeczą jest kabel sieciowy, którym podłączymy nasz moduł do ethernetu. Na czas uruchamiania najlepiej jest połączyć moduł bezpośrednio z komputerem, z którego będziemy go testować. Do tego celu potrzebny będzie kabel krzyżowy. Sposób jego wykonania przedstawiony był w EP12/2003.

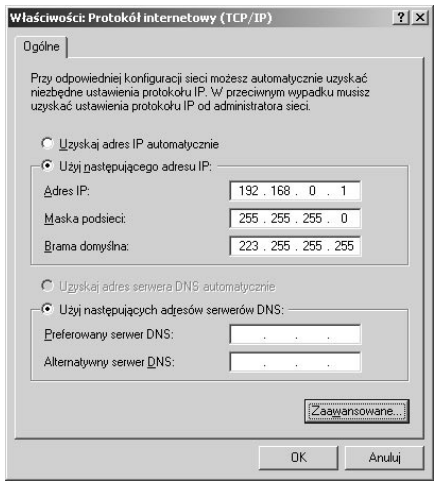
Do zaprogramowania mikrokontrolera ADuC831 niezbędny jest program ładujący WSD (*Windows Serial Downloader*) dostępny bezpłatnie ze strony producenta (*www.analog.com/microconverter*). Jest to proste i przyjazne narzędzie, działające w środowisku Windows 95/98/2k/XP. Po jego instalacji powinniśmy ustalić kilka parametrów, które ułatwią nam przyszłą pracę z modułem. Wybieramy przycisk *Configuration* i ustawiamy opcje programowania, najlepiej jak na **rys. 13**. Jako numer portu wybieramy ten, który będziemy wykorzystywać do komunikacji szeregowej z modułem. Potrzebny będzie także kabel szeregowy RS232, wykorzystywany typowo do połączenia modemu z komputerem (1:1).

Na czas testowania warto zaopatrzyć się także w dowolny program komunikacyjny. Standardowo dostępny z systemem HyperTerminal jest w zupełności wystarczający. Jeśli nie ma go w zakładce *Start-Akcesoria-Komunikacja-HyperTerminal* musimy go zainstalować z płyty CD systemu Windows. Wszystkie przykłady z projektu wykorzystują tryb 9600, n, 8, 1.

Ostatnią z rzeczy, które musimy wykonać, jest skonfigurowanie połączenia TCP/IP komputera, którym będziemy łączyć się z modułem. W prezentowanych przykładach moduł ma na stałe przypisany adres sieciowy 192.168.0.1. Aby można było się z nim bezproblemowo połączyć najlepiej jest skonfigurować parametry łącza sieciowego jak na **rys. 14**. W tym celu wybieramy *Panel sterowania-Połączenia sieciowe-*



Rys. 13. Okno konfiguracji programu WSD



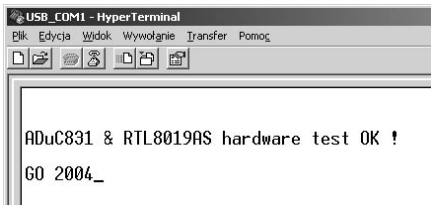
Rys. 14. Konfiguracja połączenia TCP/IP

-*Połączenie lokalne*, a następnie *Protokół internetowy (TCP/IP)*. Po zmianie parametrów może okazać się konieczne zrestartowanie komputera.

Do dzieła!

Po optycznej weryfikacji jakości lutowania i ewentualnie „przedzwonieniu" obwodu zasilania podłączamy zewnętrzny zasilacz. Jeśli posiada zabezpieczenie prądowe, to ustalamy je na 100 mA. Po włączeniu zasilania powinna zaświecić się dioda D3, sterowana przez kontroler sieci U2. Sprawdzamy obecność ważniejszych napięć (VCC, AVDD) i jeśli są prawidłowe wyłączamy zasilanie.

Czas na załadowanie pierwszego programu. Na płycie CD, załączonej do aktualnego wydania EP, znajdują się trzy skompilowane programy w postaci plików *.hex*, używane do przetestowania modułu. Pierwszy z nich, *Test1.hex*, po załadowaniu steruje okresowo diodą LED, przyłączoną do linii ACT oraz sprawdza połączenie z kontrolerem sieci. Wynik jest wysyłany na port szeregowy. Rezultat działania programu można zobaczyć na **rys. 15**. Aby wpisać program do pamięci mikrokontrolera ADuC831 zakładamy zworę na J5, podłączamy kabel między J6 a port szeregowy komputera i włączamy zasilanie. Wewnętrzny moduł POR (*Power-on-Reset*) wyzeruje mikrokontroler, zaś obecność zwory



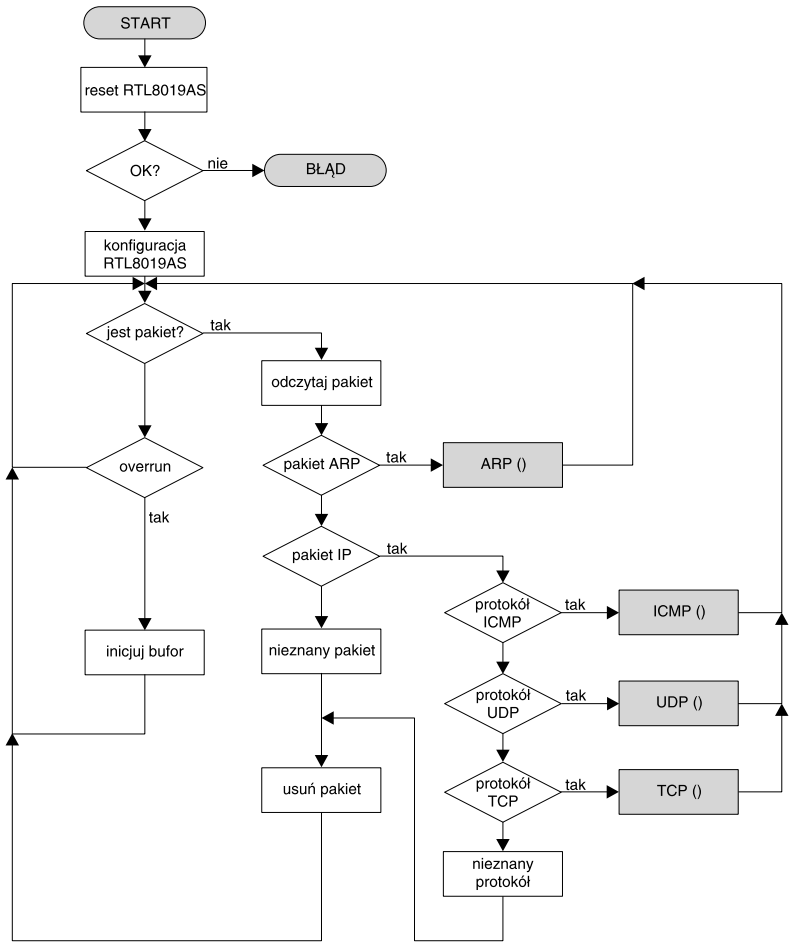
Rys. 15. Efekt działania programu Test1.hex

wprowadzi go w tryb ładowania programu. Po włączeniu programu WSD i załadowaniu wcześniej stworzonego pliku konfiguracyjnego powinniśmy zobaczyć zachętę wysyłaną przez ADuC. Jeśli tak nie jest musimy sprawdzić wszystkie składniki, począwszy od zasilania U4, jego oscylator, linię RESET, obwód układu U3 i kabel szeregowy. Jeśli dysponujemy oscyloskopem można to zrobić w miarę szybko. Przy jego braku czeka nas mozolne sprawdzenie wszystkich składników w torze RS232.

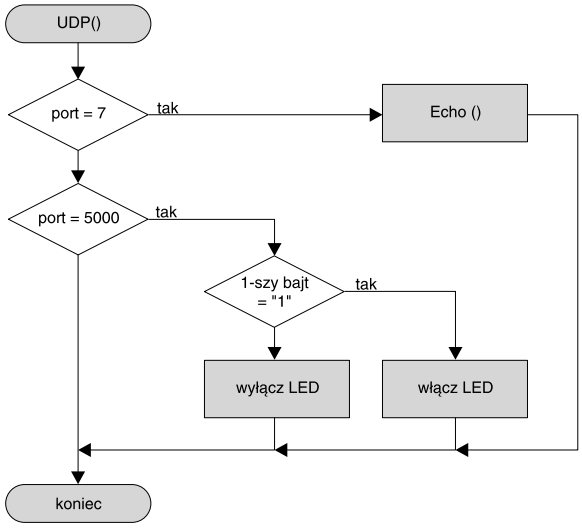
Jeśli program *Test1* wykrył obecność kontrolera, możemy uruchomić drugą aplikację testową. Program *Test2.hex* umożliwia sprawdzenie protokołów IP, ICMP, UDP, TCP oraz HTTP. Po wgraniu programu łączymy moduł z komputerem kablem krzyżowym. Zaświeci się dioda LINK oraz system Windows pokaże istniejące połączenie sieciowe.

Wybieramy *Start-Uruchom* i wpisujemy *ping 192.168.0.1*. Odpowiedź modułu na zapytanie standardową paczką o długości 32 bajtów jest testem funkcjonowania protokołów IP i ICMP oraz miarą przepustowości łącza sieciowego. Do przetestowania działania warstw TCP i HTTP najlepiej jest użyć dowolną przeglądarkę internetową. Uruchamiamy Internet Explorer i wpisujemy *http://192.168.0.1*. W odpowiedzi dostajemy zawartość prostej strony testowej, wbudowanej w program *Test2.hex*. Przy jej pomocy możemy zmienić stan diody LED oraz oczytać stan wejść analogowych i cyfrowych. Strona nie jest specjalnie „zaawansowana", ale programowanie w HTML-u nie jest silną stroną autora. Strona jest automatycznie odświeżana co 3 sekundy, poprzez dodanie w jej nagłówku dyrektywy „*refresh*".

Sprawdzenie warstwy UDP wymaga posiadania programu, który potrafi wysłać dowolną daną, pod podany adres IP, korzystając z portu o dowolnym numerze. W Internecie można znaleźć wiele takich programów. Jeden z nich - *EDTPTestPanel* - znajduje się na płycie CD. Wysłanie cyfry 1 na port 5000 powoduje zaświecenie LED, zaś każdej innej wartości jej zgaszenie. Funkcjonalność taka została zaszyta w *Test2.hex*, jako jednej z aplikacji UDP. Na porcie 7 zaimplementowano także aplikację typu *Echo*, odsyłającą zwrotnie odebrane dane. Ostatni z programów, *Test3.hex*, jest przykładem możliwości „upa-



Rys. 16. Algorytm działania programu Test2.hex



Rys. 17. Aplikacja UDP

kowania" obsługi warstw sieciowych w minimalnej wielkości zasobów mikrokontrolera. Cały program zajmuje mniej niż 4 kB kodu, z czego około 100 bajtów wypełnia strona WWW z logo ADuC831. Do jego funkcjonowania potrzeba około 130 bajtów pamięci RAM. Zarówno *Test3.hex* jak i *Test1.hex* mogą być załadowane i uruchomione na większości mikrokontrolerów rodziny 8051. Program *Test2.hex* wykorzystuje zasoby ADuC831.

Oprogramowanie

Przedstawienie szczegółów oprogramowania modułu to zadanie zdecydowanie wykraczające poza ramy niniejszego artykułu. Poniżej przedstawione więc zostaną jedynie zasady, w oparciu o które zrealizowano prezentowane przykłady. Cały program został napisany w języku C jako najbardziej efektywnym do tego celu. Szkielet obsługi modułu, zaimplementowany w przykładowym programie *Test2.hex*, przedstawia algorytm na **rys. 16**. Program rozpoczyna się od sprzętowego wyzerowania kontrolera U2 sygnałem RES. Zaraz po nim następuje *reset* programowy, inicjowany zapisem do rejestru RSTPORT (adres 0x18..0x1F). Po jego zakończeniu następuje konfiguracja kontrolera sieci - wybrany tryb TBaseT, ustawienie adresu MAC, zainicjowanie bufora kołowego (odbiorczego i nadawczego) oraz uaktywnienie odbioru ramek. Zasadnicza część programu w funkcji *main()* składa się z pętli bez końca, w której sprawdzany jest status bufora odbiorczego. Jeśli znajduje się tam jakiś pakiet, zostaje on odczytany i poddany analizie. Jeśli z jakiś powodów w buforze kołowym znajdują się nadpisane pakiety (nie obsłużone), zawartość całego bufora jest usuwana, zaś jego wskaźniki inicjowane na wartości początkowe. Rozróżnienie typu pakietu i protokołu odbywa się analogicznie do struktury modelu OSI, zaprezentowanego w pierwszej części artykułu. Obsługiwane są tylko wybrane z nich, zaś pozostałe traktowane jako nieznane i usuwane z bufora. Jak pamiętamy rozmiar pakietu może dochodzić do ponad 1500 bajtów, co zazwyczaj wynosi znacznie więcej, niż dostępna pamięć 8052. Aby sobie z tym poradzić można wszystkie operacje wykonywać na wewnętrznym buforze RTL8019AS, co jednakże powoduje wydłużenie czasu obsługi pakietu. Można także sekwencyjnie odczytywać tylko potrzebne dane i zapamiętywać wyłącznie istotne z nich (np. adres MAC i IP nadawcy). W przypadku ADuC831 dysponujemy wewnętrzną pamięcią RAM o pojemności 2 kB, którą można swobodnie wykorzystać do zapamiętania całego odebranego pakietu. Jest jednak pewien problem. Jeśli chcemy wykorzystywać rozkaz *movx A, @DPTR* do dostępu do kontrolera sieci, to przy aktywnej wewnętrznej pamięci XRAM adresy w zakresie 0...7FFh nie będą wystawiane na zewnątrz ! Spowoduje to, że U2 ulokowany pod lokacjami 0...1Fh nie będzie nigdy dostępny. Jedynym rozwiązaniem jest wówczas wyłączanie pamięci XRAM przed każdym dostępem do kontrolera (za pośrednictwem rejestru CFG831) i późniejsze jej uaktywnianie po zakończeniu cyklu. W łącznym bilansie powoduje to dłuższą obsługę, niż przy prostym sterowaniu liniami I/O. Problem ten oczywiście można łatwo rozwiązać dodając jeden inwerter na linii CS. W przyjętym schemacie zrezygnowano jednak z takiej koncepcji celem maksymalnego uproszczenia konstrukcji modułu.

UDP ()

W programach *Test2* i *Test3* zaimplementowano aplikację *UDP()*, której schemat funkcjonalny zawiera **rys. 17**. Po wykryciu numeru portu przeznaczenia o wartości 7 (*Echo*) odsyłany jest zwrotnie cały odebrany pakiet. Jeśli port ma wartość 5000, to w zależności od wartości pierwszego bajtu danych UDP zmieniany jest stan diody LED na linii ACT.

HTTP ()

Warstwa TCP posiada jedną aplikację, obsługiwaną na porcie o wartości 80. Jeśli pojawią się jakiekolwiek dane, odebrane przez TCP, wywoływana jest obsługa *HTTP()*. W przypadku programu Test3.hex zwrotnie odsyłany jest ciąg danych: *HTTP/1.0 200 OK\r\nContent-type: text/html\n\n<html>\n<body>\n"ADuC831"\n\n</body>\n</html>\n*, stanowiący zawartość prostej strony internetowej. W przypadku programu *Test2.hex* dodatkowo wstawiane są sformatowane do postaci ASCII wartości napięć wejść analogowych i stany końcówek I/O0..3. Taki sposób tworzenia *dynamicznych* stron WWW wymaga zazwyczaj umieszczania specjalnych *tagów* wewnątrz kodu HTML, które podczas ładowania są podmieniane na aktualne wartości kodowanych sygnałów.

Na zakończenie

W zamyśle autora artykuł miał przybliżyć na pozór trudne zagadnienia związane z obsługą sieci ethernet. Sądząc po konstrukcji modułu nie jest to jednak wcale skomplikowane. Nieco więcej zachodu wymaga obsługa programowa, którą jednak można sprowadzić do bardzo niewielkiego zakresu. Uzyskanie funkcjonalnego stosu TCP/IP wymaga tylko kilku kB pamięci, co po uzupełnieniu o własne pomysły pozwala na stworzenie bardzo efektywnego urządzenia. Jako kierunki rozwoju oprogramowania warto polecić oprogramowanie protokołu DHCP, pozwalającego dynamicznie podłączać moduł do sieci oraz FTP, przy pomocy którego można łatwo aktualizować dane między modułem i jego otoczeniem sieciowym. Stronę sprzętową można natomiast uzupełnić o interfejs PoE, który można oprzeć np. o układ TPS2370.

Grzegorz Oleszek
go@savo.pl