EASY TCP/IP

Instrukcja Obsługi

Wersja 1.00

Wstęp

Wraz upowszechnieniem się Internetu i technologii ADSL w wielu krajach, możemy prowadzić interesy niemal z każdego miejsca w świecie. Bankowość internetowa, wysyłanie i odbieranie poczty elektronicznej, zakupy przez Internet; wszystko to jest dostępne z dowolnego miejsca gdzie jest dostęp do sieci Internet.

Mimo iż wielu użytkowników nadal używa dostępu za pomocą modemów, w przeciągu kilku lat gniazdo sieci Internet będzie tak samo powszechne jak gniazdo sieci energetycznej. Aktualnie wielu już użytkowników posiada małą sieć typu LAN (*Local Area Network*) składającą się z dwóch lub więcej komputerów. I tylko krok dzieli ich od światowej sieci.
Wiele urządzeń domowego użytku w przyszłości będzie wyposażonych w charakterystyczne gniazdo RJ-45. Czy wyobrażasz sobie piekarnik który może otrzymywać przepisy kulinarne z internetowej bazy, telewizor kontrolowany zdalnie przez komputer PC lub magnetowid programowany za pomocą prostego sieciowego interfejsu?
Wiem, pytanie to brzmi nieco przesadnie, lecz tylko przyszłość potrafi na nie odpowiedzieć.

Komputery PC są stosunkowo tanie i potrafią pracować z ogromnymi aplikacjami, lecz mają podstawową wadę: muszą być stale zasilane i pobierają tym samym ogromną ilość energii.
Małe systemy mikroprocesorowe nie posiadają tejże wady gdyż zazwyczaj nie pobierają aż takiej ilości energii. Jednak do czasu gdy firma Wiznet nie opracowała układu W3100A, ciężko było połączyć mikrokontroler z siecią Internet.

Widziałem sporo układów przeznaczonych do łączenia mikrokontrolerów za pomocą sieci Ethernet, lecz wszystkie zużywały sporo zasobów (kod programu, pamięć) by przeprowadzić nawet proste zadania.
Układ W3100A, produkowany przez firmę Wiznet (www.i2chip.com) zmieni tą sytuację. Układ ten oferuje 4 tzw. gniazda oraz posiada własną pamięć na odczytane i nadawane dane. Ponieważ układ W3100A jest także produkowany w formie specjalnego modułu (IIM7000), który może być używany przez hobbystów (nie trzeba lutować układów SMD), MCS Electronics zdecydowało się wspomóc ten znakomity układ za pomocą produkowanych przez siebie kompilatorów BASIC'a: BASCOM-AVR oraz BASCOM-8051. W tym celu opracowano kompletny system uruchomieniowy: **Easy TCP/IP**. Nie martw się jeśli nie rozumiesz protokołu TCP/IP gdyż pakiet **Easy TCP/IP** – czyli płytka prototypowa i dołączone do niej oprogramowanie - został zaprojektowany dla początkujących.

Jak rozpocząć?

By rozpocząć pracę z pakietem **Easy TCP/IP**, musisz posiadać choćby małą sieć. Minimum to komputer PC z zamontowaną kartą sieciową.
Znajomość języka BASIC oraz(lub) BASCOM Basic także będzie potrzebna.

Podstawy TCP/IP

Podstawowym przeznaczeniem sieci jest wymiana danych pomiędzy wieloma terminalami i komputerami. Jeden z najpopularniejszych aktualnie protokołów sieciowych to TCP/IP. TCP to skrót od *Transmission Control Protocol* (protokół kontroli transmisji) a IP od *Internet Protocol*.

Istnieje wiele warstw w protokołach sieciowych. Jedną z nich jest warstwa sprzętowa (twoja karta sieciowa oraz płytka **Easy TCP/IP**), drugą zaś warstwa programowa (aplikacje korzystające z danego protokółu).

Zarówno TCP/IP jak i RS-232 to protokoły oparte na transmisji szeregowej. W komputerze PC, w miejsce portu szeregowego używamy karty sieciowej. Na zewnątrz możemy ją dołączyć do samodzielnego urządzenia: przykładowo **Easy TCP/IP**, TINI, itd.
W porównaniu do RS-232, TCP/IP jest o wiele bardziej zaawansowanym protokołem. Dlaczego potrzebny jest aż taki skomplikowany protokół?

Protokół to forma porozumienia pomiędzy dwoma lub więcej urządzeniami i dlatego są w stanie się porozumieć. Dla RS-232, najważniejsze są: szybkość transmisji w bodach, ilość bitów danych, itd. Bez znajomości tych parametrów obydwa urządzenia nie potrafiły by się połączyć lub też wymiana danych była by obarczona błędami.

Można by długo dyskutować jak działa TCP/IP. Ja ze swej strony przedstawię tylko te aspekty które są niezbędne. W TCP/IP do komunikacji pomiędzy dwoma aplikacjami używamy tzw. „gniazd". Gniazdo można porównać z kanałami w języku Qbasic lub Visual Basic. Gdy użyjesz: `OPEN „plik" FOR BINARY AS #1`, zostanie otwarty kanał za pomocą którego można będzie odczytywać i zapisywać dane z(do) pliku o nazwie `plik`. Podany numer kanału: `#1` jest wszystkim co musisz znać by zidentyfikować plik do którego trafiać będą odebrane dane, lub z którego dane będą pobierane.
W nomenklaturze TCP/IP kanały transmisyjne są właśnie nazywane gniazdami. Aby skomunikować się z klientem bądź serwerem musisz najpierw otworzyć gniazdo.

Zdefiniujmy teraz iż: **serwer** – to proces który oczekuje na dołączenie się klienta; **klient** – to aplikacja która żąda połączenia się z serwerem.
Przykładowo jeśli odbierasz pocztę elektroniczną zostajesz połączony z tzw. serwerem pocztowym. Twój program za pomocą którego odbierasz czy nadajesz pocztę zwie się klientem poczty elektronicznej.
Większość serwerów pozwala na dołączenie się kilku klientów naraz. W tym samym czasie gdy ty odbierasz pocztę, kilku innych użytkowników może także swoją pocztę odbierać.

Skoro funkcjonuje tyle serwerów, to w jaki sposób możemy określić z którym z nich chcemy się połączyć? Robimy to za pomocą tzw. **adresu IP** (zwanego także numerem IP). Adres ten składa się z 4 bajtów oddzielonych za pomocą kropek:
192.168.0.10
Ponieważ każda z czterech części adresu reprezentuje 8 bitów, liczby są zapisywane w postaci normalnych liczb dziesiętnych. (Aktualnie by zapamiętać numery IP stosowana jest zmienna typu LONG – 4 bajty).

Jeśli używasz przeglądarki WWW także podajesz numer IP. Przykładowo:
http://64.77.35.38

skieruje cię do mcselec.com. Ponieważ ludzie nie mają pamięci do liczb – zwłaszcza długich – została stworzona tzw. usługa DNS (*Domain Name Service*). Zamienia ona w tle nazwę domeny „mcselec.com" na właściwy adres IP.
Każdy z adresów IP musi być niepowtarzalny. Dlatego też podając 64.77.35.38 zawsze będziesz skierowany do mcselec.com.

Przedrostek **http://** jaki wpisujesz w przeglądarce internetowej podpowiada jej, że będziesz używał protokołu HTTP. Standardowo serwery WWW używają domyślnego portu o numerze 80 na potrzeby HTTP.
Ale właściwie co to jest ten port?

Możesz wyobrazić sobie port jako departament jednego z urzędów. Adres urzędu jest zawsze taki sam dla poszczególnych jego departamentów, lecz by zaadresować list do konkretnego departamentu musisz podać jego nazwę.
Nie pomyl portów z portami w które wyposażony jest mikrokontroler! Porty w TCP/IP należy raczej kojarzyć z podadresami.
Numer portu to liczba typu WORD zawierająca się w zakresie 0 – 65535. Wiele z portów jest używanych przez konkretne usługi (lista tych usług znajduje się na końcu tego dokumentu).
Serwery WWW używają zazwyczaj portu 80, a serwery poczty POP3 używają portu o numerze 110.

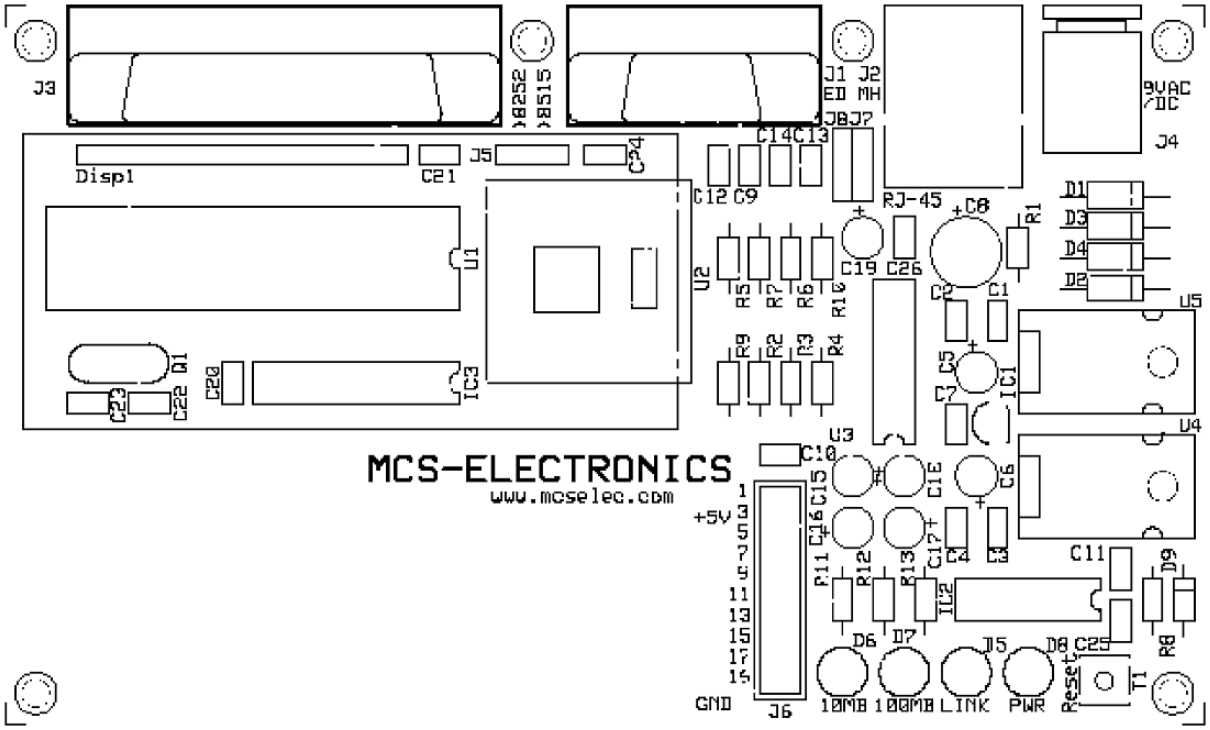
Oczywiście możesz także utworzyć serwer WWW który będzie nasłuchiwał portu o numerze 5000. W tym wypadku musisz poinformować klienta (przeglądarkę), że powinna połączyć się z portem o tym właśnie numerze. Przykładowo:
http://64.77.35.38:5000/index.htm
Dodatkowy parametr **:5000** bezpośrednio za adresem IP to właśnie numer portu serwera z którym chcesz się połączyć.
By dopełnić opisu, nie tylko serwer używa portów – także klient używa portu. Nazywamy go wtedy portem lokalnym.

Podsumowując: aby zainicjować połączenie zawsze potrzebne będzie otwarte gniazdo oraz zawsze będzie trzeba określić czy pracujemy w trybie serwera czy klienta.
Po nawiązaniu połączenia można wysyłać lub odbierać dane. Jakie to będą dane to zależy od używanego protokołu. Przykładowo protokół HTTP to protokół oparty na przesyłaniu tekstu.
Wysyłamy oraz odbieramy czysty tekst ASCII. Możesz także nawiązać połączenie z serwerem który odbiera i nadaje dane binarne (np. FTP).

Czas zacząć!

Teraz czas zaczynać. Wszystko się powoli wyjaśni jeśli rozpoczniesz próby praktyczne.

Rozpocznij od zmontowania płytki. Najpierw zmontuj cześć zasilającą oraz przylutuj wszystkie elementy bierne. Rozkład elementów przedstawia poniższy rysunek:



Szczegółową listę elementów umieszczono poniżej. W każdym wypadku zalecane jest aby pod układy scalone i moduł IIM7000 zastosować podstawki.

Element	*Opis*
R1	rezystor 1kΩ
R2, R3, R4, R9	rezystor 51Ω, 1%
R5, R6, R7	rezystor 150Ω
R8	rezystor 10kΩ
R10	rezystor 22Ω. Opcjonalny, przy podświetlanym LCD
R11, R12, R13	rezystor 1kΩ
D1, D2, D3, D4	1N4001. Uwaga na polaryzację!
D9	dioda Zenera 3,3V. Uwaga na polaryzację!
C1, C2, C3, C4, C7, C9, C10, C11, C13, C14, C20, C21, C24, C25, C26	kondensator 100nF, najlepiej MKT
C22, C23	kondensator ceramiczny 22 pF
C12	kondensator ceramiczny 100 pF
Podstawki pod układy scalone	14 pin, 16 pin, 20 pin oraz 40 pin
Podstawka pod rezonator	opcjonalnie, jeśli zajdzie potrzeba zmian rezonatora.
U5	LM7805 lub odpowiednik.

Element	*Opis*
U4	LD2517V33 lub LD33V albo inny stabilizator 3,3V w obudowie TO-220.
Żeńskie, dwurzędowe, wąskie gniazdo goldpin dla U2	opcjonalnie, jeśli moduł ma być używany także w innym urządzeniu. **Uwaga!** Używany jest rozstaw 2mm! Dlatego, iż moduł IIM7000 posiada końcówki rozstawione w rastrze 2mm, nie 2,54mm.
Żeńskie dwurzędowe gniazdo goldpin 20x2 dla J6	opcjonalnie.
Jednorzędowy męski goldpin dla J7, J8, J5	
Jednorzędowe gniazdo goldpin dla LCD	opcjonalnie, jeśli zaistnieje potrzeba użycia wyświetlacza LCD.
D5, D6, D7	LED, żółta, 3mm. Okrągły punkt lutowniczy to anoda (dłuższa końcówka LED).
D8	LED, zielona, 3mm. Okrągły punkt lutowniczy to anoda (dłuższa końcówka LED).
C15, C16, C17, C18 , C19	kondensator 1 µF/35V. Uwaga na polaryzację!
C5, C6	kondensator 10 µF/35V. Uwaga na polaryzację!
C8	kondensator 220 µF/35V. Uwaga na polaryzację!
T1	przycisk reset (mikrostyk)
J3	DB-25 męski do druku
J1	DB-9 żeński do druku
J2	gniazdo modułowe RJ-45 do druku
J4	gniazdo zasilające
IC1	ZSM560C, TO-92. Jest scalony układ generacji sygnału reset wraz z monitorem zasilania.

Jeszcze nie wkładaj układów scalonych do podstawek.

Po przylutowaniu wszystkich elementów, sprawdź płytkę pod silnym światłem i wyeliminuj ewentualne zwarcia ścieżek.
Podłącz teraz zasilanie. Napięcie zasilające musi zawierać się w granicach 6-15V DC.

Polaryzacja nie ma tutaj znaczenia gdyż używany jest mostek prostowniczy. Dioda sygnalizująca zasilanie powinna teraz świecić.
Zmierz napięcia na wyjściach U4 oraz U5. Napięcia muszą być następujące: 5V na końcówce 3 układu U5 oraz 3,3V na końcówce 2 układu U4.

Uwaga! Stabilizator 3,3V ma inaczej ułożone wyprowadzenia niż układy serii 78xx.

Jeśli napięcia nie są prawidłowe, odłącz zasilanie i sprawdź połączenia na płytce. Najlepiej by wkładka radiatorowa stabilizatora 3,3V była wewnętrznie połączona z wyprowadzeniem masy. Jeśli posiadasz inny stabilizator, musisz użyć podkładki izolacyjnej i tulejki na śrubę mocującą.

Jeśli napięcia są prawidłowe, odłącz zasilanie i włóż układy CMOS (74HC00 i 74HC573) oraz bufor MAX-232. Włóż także moduł IIM7000. Uważaj by nie zamontować go odwrotnie. Rezonator kwarcowy modułu IIM7000 musi odpowiadać pozycji narysowanej na opisowej stronie płytki.

Włóż zworkę J5. Pozycja zworki zależy od użytego typu mikrokontrolera – 8051 lub AVR. Włóż także mikroprocesor. Jeśli planujesz używać procesora z rodziny AVR użyj układu ATMega161L.

Włóż także zworki J7 oraz J8 odpowiedzialne za konfigurację portu komunikacji szeregowej. Zworka J7 musi być włożona na pozycji 1-2, a J8 na pozycji 2-3. Wtedy zarówno port COM komputera jak i port na płytce Easy TCP/IP będą miały linie TxD i RxD dołączone do tych samych końcówek. Do połączenia płytki z komputerem należy użyć kabla typu null-modem, gdyż kabel taki posiada połączenia krzyżowe.
Kabel programujący (LPT) musi mieć połączenia 1:1. Większość kabli przedłużających LPT jest odpowiednia. Należy tylko pamiętać aby był to kabel dobrej jakości i nie był zbyt długi.

Podłącz także kabel sieciowy – skrętkę UTP lub STP (Z przeplotem lub bez, zależnie od tego czy łączysz się za pośrednictwem HUB-a lub Switch-a, czy też bezpośrednio).

Uruchom BASCOM-AVR oraz wybierz programator Sample Electronics. Wybierz także właściwy adres portu LPT. W BIOSie port drukarki powinien być ustawiony jako EPP.

Włącz zasilanie i załaduj program przykładowy `tcpip.bas`. Skompiluj go i zaprogramuj mikrokontroler.
Program przykładowy `tcpip.bas` ustawia adres IP płytki **Easy TCP/IP**. Powinno być teraz możliwe wysłanie pakietu kontrolnego z podłączonego komputera PC!

Użyj polecenia PING z wiersza poleceń:

```
C:\WINDOWS\>ping 192.168.0.8
```

Dioda sygnalizująca 10 Mb powinna zamrugać i powinieneś otrzymać odpowiedź. Świadczy to o poprawnej pracy płytki i jest ona gotowa by wykonać kilka zaawansowanych zadań.

Uwaga! W tekście tym domyślnym adresem IP płytki **Easy TCP/IP** jest 192.168.0.8. Jeśli chcesz użyć innego adresu musisz go zmienić w programie (instrukcja **CONFIG TCPIP**). Jako adres IP komputera używany jest 192.168.0.3. Jeśli twój komputer ma ustawiony inny adres IP także powinieneś zmienić ten adres w programach.

Jeśli próby z poleceniem PING nie przyniosły rezultatu, powinieneś sprawdzić czy twoja karta sieciowa jest skonfigurowana poprawnie. W tym celu możesz skorzystać z informacji zawartych w następnych dwóch sekcjach.

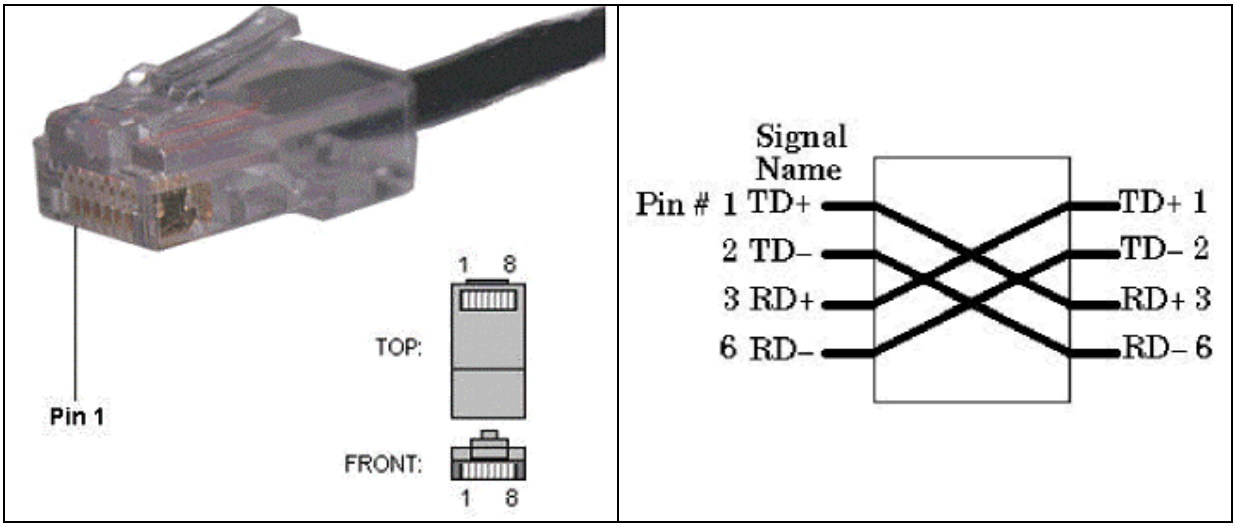
Konfiguracja sieci

Jest wiele możliwości by utworzyć prostą sieć. Wystarczy że zainstalujesz kartę sieciową w twoim komputerze PC. Karta ta może być potem połączona z płytką **Easy TCP/IP PCB** za pomocą kabla z tzw. przeplotem.

Możesz także zakupić HUB-a lub Switch-a i połączyć do niego kartę sieciową twojego komputera(-ów). Płytka **Easy TCP/IP** musi być połączona z HUB-em lub Switch-em z pomocą normalnego kabla sieciowego UTP lub STP zakończonego wtykami RJ-45.

HUB to coś w rodzaju rozdzielacza RJ-45. Używany jest do wzajemnego połączenia kilku kart sieciowych i tym samym do stworzenia małej sieci LAN. Switch to bardziej inteligentne urządzenie w porównaniu z HUB-em. Dodatkowo kieruje on transmisją tak, aby sieć działała sprawniej.

Jeśli nie posiadasz HUB-a lub Switch-a a chcesz połączyć płytkę **Easy TCP/IP** bezpośrednio z karta sieciową komputera musisz użyć kabla z tzw. przeplotem. Możesz go kupić w każdym sklepie ze sprzętem komputerowym lub wykonać samemu:

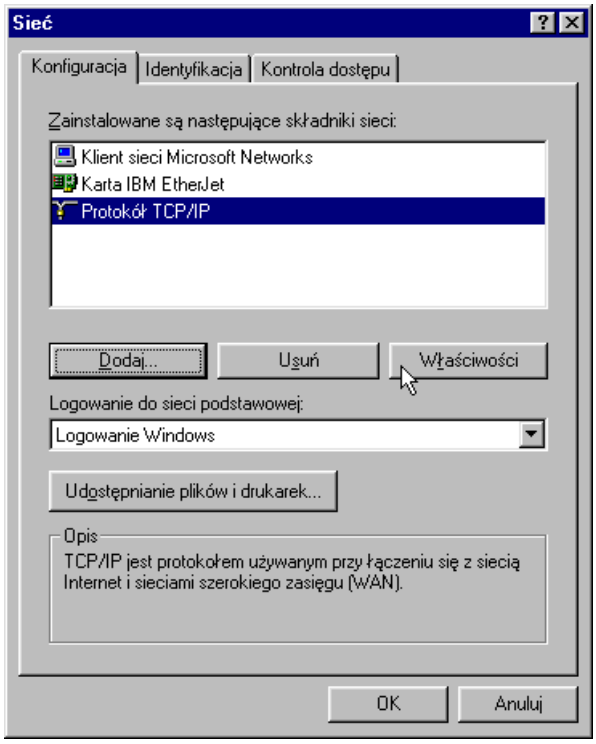


Powyższe rysunki pokazują prawidłowe-minimalne połączenia dla kabla 10Base-T z przeplotem.

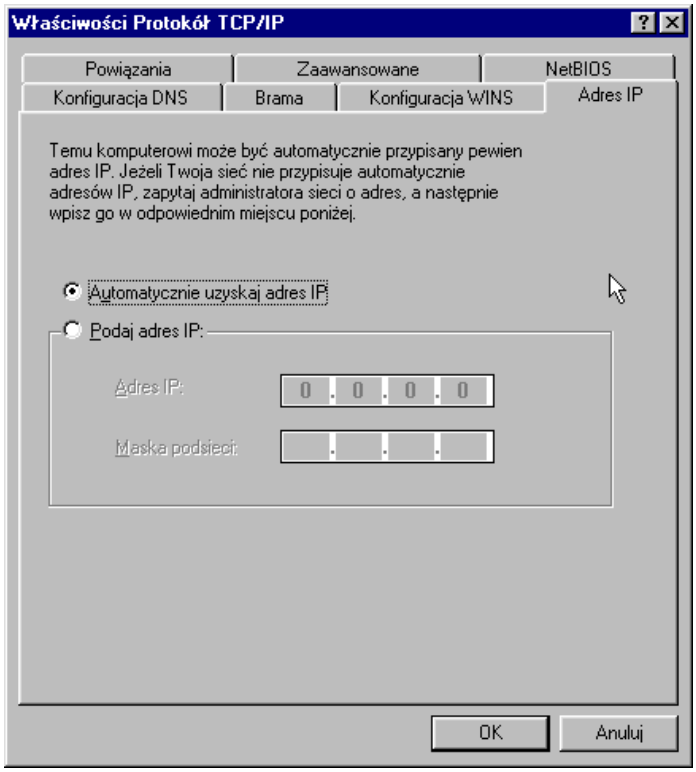
Użyj kabla z przeplotem wyłącznie w sytuacji gdy płytka Easy TCP/IP będzie łączona bezpośrednio z kartą sieciową twojego komputera!

Konfiguracja sieci w komputerze.

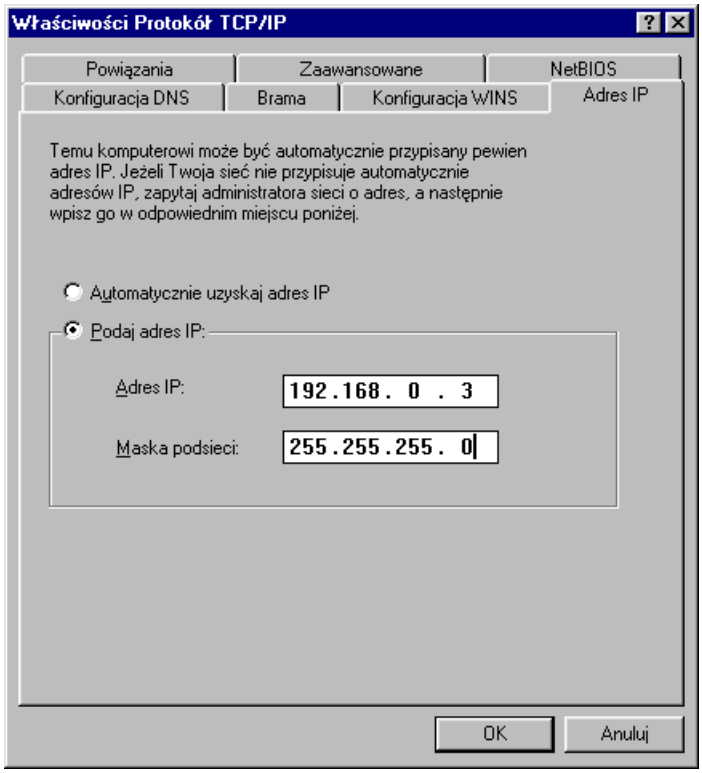
Wybierz z **Panelu sterowania** aplet **Sieć**. Powinno się pojawić takie oto okno:



Pokazane będą składniki sieci, m.in. nazwa karty sieciowej, zainstalowane protokoły itd. Wybierz **Protokół TCP/IP** oraz kliknij na **Właściwości**. Powinno się pojawić następne okno:



Jeśli używasz serwera DHCP w celu uzyskiwania adresu IP komputera, nie potrzebujesz niczego robić. Jeśli nie używasz DHCP, okno powinno wyglądać tak:



Pole **Adres IP** może być puste. Wpisz zatem **192.168.0.3** w polu **Adres IP** oraz **255.255.255.0** w polu **Maska podsieci**.

Po wciśnięciu **OK**, w zależności od używanego systemu operacyjnego możesz być poproszony o ponowne uruchomienie komputera.
Po opcjonalnym restarcie, uruchom **Tryb MS-DOS**(**Linię poleceń**) i wpisz polecenie IPCONFIG.

Spowoduje to wyświetlenie właściwości IP:

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>ipconfig

Windows 2000 IP Configuration

Ethernet adapter Local Area Connection 2:

Connection-specific DNS Suffix  . :
IP Address. . . . . . . . . . . . : 192.168.0.3
Subnet Mask . . . . . . . . . . . : 255.255.255.0
```

Możesz teraz spróbować wysłać pakiet kontrolny do twojego komputera. Wpisz zatem w linii poleceń:

```
C:\WINDOWS>ping 192.168.0.3
```

W rezultacie powinieneś otrzymać następujące informacje:

```
Badanie 192.168.0.3 z użyciem 32 bajtów danych:

PING: 192.168.0.3: bajtów=32 czas<10ms TTL=128
PING: 192.168.0.3: bajtów=32 czas<10ms TTL=128
PING: 192.168.0.3: bajtów=32 czas<10ms TTL=128
PING: 192.168.0.3: bajtów=32 czas<10ms TTL=128

Statystyka badania dla 192.168.0.3:
   Pakiety: Wysłane = 4, Odebrane = 4, Utracone = 0 (0% utracono)
Szacunkowy czas błądzenia pakietów w milisekundach:
   Minimum = 0ms, Maksimum = 0ms, Średnia = 0ms
```

Polecenie PING wysyła pakiet danych pod określony adres IP. Karta sieciowa powinna zwrócić te dane z powrotem. PING używany jest w celu określenia czy serwer/klient jest on-line oraz do pomiaru opóźnień w transmisji.

Sewer DHCP to specjalny serwer, który przypisuje adresy IP klientom w sieci. Jest on bardzo pomocny; inaczej wszystkie adresy IP komputerów należałoby określać ręcznie. Podczas startu komputera, kontaktuje się on z serwerem DHCP i pobiera automatycznie adres IP. W zależności od ustawień serwera DHCP adres IP jest zawsze taki sam (adres statyczny) lub może się zmieniać po każdym restarcie komputera (adres dynamiczny). Gdy DHCP nie jest używany, to może się zdarzyć sytuacja, że w sieci pojawią się dwa komputery o tym samym adresie. Spowoduje to niewątpliwie konflikt. I to jest właśnie powód, że użycie serwera DHCP jest rekomendowane dla uzyskiwania adresu IP. Nie pozwoli on na pojawienie się dwóch tych samych adresów.

Adresy IP 192.168.0.0 – 192.168.0.255 to specjalny zakres adresów. Nie są one używane w sieci Internet.
Ponieważ zakres ten nie jest używany w ogólnoświatowej sieci, jest używany w sieciach LAN - które także mogą być podłączone do sieci Internet. Jeśli nie łączysz się z siecią Internet przez LAN możesz użyć dowolnego z adresów IP. Lecz zalecane jest jednak używanie adresu w postaci 192.168.0.xxx. Jeśli adres 192.168.0.3 został przypisany jako adres internetowy, powinieneś skontaktować się z klientem/serwerem przez Internet, nie jako z PC/urządzeniem w sieci LAN.

W następnej sekcji zawarto informacje na temat poszczególnych instrukcji i funkcji związanych z TCP/IP, które są dostępne w BASCOM AVR w wersji 1.11.7.2 lub nowszej, wzbogaconej o bibliotekę `TPCIP.LIB` dostarczoną razem z płytką **Easy TPC/IP**.

Biblioteka TPCIP

Biblioteka BASCOM TCP/IP pozwala na używanie „internetowego" układu W3100A firmy Wiznet www.i2chip.com

Układ W3100A jest zamontowany na module IIM7000. Możesz go także kupić oddzielnie oraz użyć we własnym projekcie. Poza modułem IIM7000, dostępny jest także moduł IIM7010 który posiada zamontowane sieciowe gniazdo RJ-45.
Zajrzyj do dokumentacji w plikach PDF, która jest dostarczana razem z płytką **Easy TCP/IP**.

Biblioteka `tcpip.lib` zawiera kod maszynowy dla następujących instrukcji i funkcji języka BASCOM Basic:

CONFIG TCPIP

BASE64DEC
CLOSESOCKET
GETDSTIP
GETDSTPORT
GETSOCKET
SOCKETCONNECT
SOCKETLISTEN
SOCKETSTAT
TCPWRITE
TCPWRITESTR
TCPREAD

UDPWRITE
UDPWRITESTR
UDPREAD

Funkcje dotyczące protokołu UDP działają podobnie jak funkcje związane z protokołem TCP. Protokół UDP to protokół bezpołączeniowy. Oznacza to, że nie pracujesz w trybie nasłuchiwania oraz nie łączysz się z żadnym serwerem.
Wystarcza tylko że otworzysz gniazdo funkcją **GETSOCKET**. Potem możesz wysyłać i odbierać dane.

Normalnie adres IP oraz numer portu podajesz przy nawiązywaniu połączenia. Korzystając jednak z UDP nie łączysz się, więc adres i port musisz podać gdy wysyłasz dane. To jest właśnie główna różnica pomiędzy funkcjami **TCPWRITE** a **UDPWRITE**. UDP jest szybszy lecz także mniej niezawodny w stosunku do TCP.

Ponieważ przy UDP nie ma serwera oraz klienta, fakt odbierania danych możesz stwierdzić tylko przez sprawdzenie liczby odebranych bajtów. Jeśli zaś wysyłasz dane, nie będziesz nigdy wiedział czy rzeczywiście doszły one i czy w tym samym porządku. Wysłanie przykładowo kolejno 3 pakietów za pomocą UDP, nie oznacza że doszły one w tym samym porządku do adresata.
Dlatego radzę ci abyś używał TCP/IP tam gdzie tylko to możliwe.

Uwaga! Biblioteka TCPIP nie jest standardowym składnikiem BASCOM AVR. Przed pierwszym użyciem należy skopiować pliki `easytcp.lib` oraz `easytcp.lbx` do katalogu `lib` umieszczonego w katalogu z programem BASCOM AVR.

Opis instrukcji i funkcji biblioteki TCPIP

CONFIG TCPIP

Przeznaczenie:
Konfiguruje scalony układ komunikacji TCP/IP - W3100A.

Składnia:
CONFIG TCPIP = INT0 | INT1 , MAC = *numer_MAC* **, IP =** *adres_IP* **, SUBMASK =** *maska* **, GATEWAY =** *adres_bramy* **, LOCALPORT =** *nr_portu* **, TX =** *tx* **, RX =** *rx*

Opis:
Instrukcja służy do skonfigurowania układu W3100A oraz procedury współpracy z tym układem. Ponieważ układ generuje przerwania należy określić które przerwanie jest podłączone do układu W3100A. Dla płytki prototypowej **Easy TCP/IP** należy podać INT0.

Parametr MAC= określa unikalny numer jednoznacznie identyfikujący układ W3100A. Jest to odpowiednik unikalnego numeru każdej karty sieciowej Ethernet.
Każdy z układów podłączony do sieci musi mieć inny numer. Numer ten składa się z ciągu 6 liczb (od 0 do 255) oddzielonej kropkami. Przykładowo: 123.00.12.34.56.78

Parametr IP= określa adres IP układu, który także musi być unikalny w sieci. Jeśli posiadasz sieć LAN możesz użyć: 192.168.0.10, gdyż wszystkie adresy rozpoczynające się od 192.168.0.x są zarezerwowane dla sieci LAN i nie występują jako adresy IP w sieci Internet.
Parametr SUBMASK= określa tzw. maskę podsieci dla układu W3100A. W większości przypadków maska ta jest ustawiona na 255.255.255.0
Parametr GATEWAY= określa adres domyślnej bramy. Adres ten można określić wpisując w linii poleceń systemu Windows polecenie: `C:\>ipconfig` Rezultat może wyglądać tak:

```
Windows 2000 IP Configuration
Ethernet adapter Local Area Connection 2:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 192.168.0.3
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.0.1
```

W tym przypadku adres ten to 192.168.0.1.

Wartość (typu Word) przypisana parametrowi LOCALPORT= jest przepisywana do wewnętrznej zmiennej `LOCAL_PORT`. Zobacz opis funkcji **GETSOCKET**. Domyślnie możesz podać 5000.

Za pomocą parametru TX= można określić liczbę oraz wielkość bufora transmisyjnego. Układ W3100A posiada 4 tzw. *gniazda* (socket). Dla każdego gniazda przypisano osobne

dwa bity w bajcie. Stan 00 spowoduje określenie bufora na 1024 bajtów, 01 - 2048 bajtów, 10 - 4096 bajtów a 11 - 8192 bajtów. Dwa młodsze bity określają wielkość gniazda 0, a dwa najstarsze - gniazda 3. Przykładowo TX = &B01010101 spowoduje, iż wielkość bufora dla każdego gniazda będzie wynosić 2048 bajtów.
Ponieważ bufor transmisyjny ma tylko 8KB, można go podzielić na 4 części po 2048 bajtów (tak jak wyżej). Można oczywiście określić wielkość bufora na 8KB, lecz w tym wypadku możliwe jest użycie tylko gniazda 0 i parametr powinien TX= powinien wyglądać tak: TX = &B11.
Parametr RX= określany jest tak samo jak TX= z tą różnicą że dotyczy bufora odbiorczego.

Więcej informacji można znaleźć przeglądając notę aplikacyjną układu W3100A.

Instrukcja **CONFIG TCPIP** może być użyta tylko raz. Przedtem należy włączyć globalny system przerwań, gdyż wykorzystywane są przerwania INT0/1. Instrukcja automatycznie inicjalizuje układ W3100A.

Po wykonaniu **CONFIG TCPIP** można już wysyłać pakiety kontrolne poleceniem ping

Przykład:

```
Config Tcpip = Int0 , Mac = 00.00.12.34.56.78 , Ip = 192.168.0.8 , Submask =
255.255.255.0 , Gateway = 192.168.0.1 , Localport = 1000 , Tx = $55 , Rx =
$55

' Teraz użyj polecenia PING z linii poleceń by wysłać pakiet kontrolny i
sprawdzić czy układ odpowiada:
'  PING 192.168.0.8
' Możesz użyć także programu EasyTcp.
```

BASE64DEC()

Przeznaczenie:
Wykonuje konwersję danych zapisanych w formacie Base-64 na postać oryginalną.

Składnia:
rezultat **= BASE64DEC (** *dane* **)**

gdzie

rezultat	ciąg znaków do którego wpisane będą przetworzone dane,
dane	ciąg znaków z zakodowanym ciągiem.

Opis:
Format Base-64 nie jest żadnym z protokołem szyfrującym. Przesyła on dane w postaci ciągu 7-bitowych znaków ASCII. MIME, serwery Web oraz reszta serwerów i klientów w Internecie używa kodowania Base-64

Przeznaczeniem funkcji **BASE64DEC** jest wyłącznie dekodowanie informacji. Wspomaga ona przeprowadzenie operacji autoryzacji do serwera Web. Gdy serwer pyta o autoryzację, klient powinien wysłać nazwę i hasło użytkownika w postaci niezaszyfrowanej, zapisanej w formacie Base-64.
Zakodowane ciągi w formacie Base-64 składają się z pary 4 bajtów. Te 4 bajty reprezentują 3 normalne bajty.

CLOSESOCKET

Przeznaczenie:
Zamyka połączenie z gniazdem.

Składnia:
COLSESOCKET *nr_gniazda*

gdzie:

nr_gniazda	jest to numer gniazda z zakresu **0** - **3** które należy zamknąć. Jeśli gniazdo zostało już zamknięte instrukcja nie wykonuje żadnych operacji.

Opis:
Należy zamknąć gniazdo jeśli zostanie odebrany komunikat `SOCK_CLOSE_WAIT`. Można także zamknąć gniazdo jeśli wymaga tego używany protokół komunikacyjny. Komunikat `SOCK_CLOSE_WAIT` powinien być odebrany w momencie gdy serwer zamyka połączenie.

Użycie instrukcji **CLOSESOCKET** powoduje zamknięcie aktualnego połączenia.

Uwaga! Nie jest wymagane oczekiwanie na komunikat `SOCK_CLOSE_WAIT` by zamknąć połączenie ze strony klienta.

Po zamknięciu gniazda należy użyć instrukcji **GETSOCKET** by ponownie skorzystać z tego gniazda.

GETDSTIP()

Przeznaczenie:
Zwraca adres IP połączonego użytkownika sieci.

Składnia:
rezultat **= GETDSTIP(** *gniazdo* **)**

gdzie:

rezultat	zmienna typu Long gdzie wpisany będzie odczytany adres IP,
gniazdo	zmienna lub stała określająca numer gniazda (**0** – **3**).

Opis:
Jeśli system pracuje jako serwer, może być wymagane odczytanie adresu IP połączonego klienta. Można to wykorzystać do logowania tylko wybranych klientów czy jako element systemu zabezpieczeń.

Uwaga! Bajt MSB adresu IP jest zwracany w bajcie LSB zmiennej Long.

Przykład:

```
Dim L As Long
L = GetdstIP(i)        ' pobierz adres IP dla gniazda o numerze i
```

GETDSTPORT()

Przeznaczenie:
Zwraca numer portu połączonego użytkownika sieci.

Składnia:
rezultat = **GETDSTPORT(** *gniazdo* **)**

gdzie:

rezultat	zmienna typu Word gdzie wpisany będzie odczytany numer portu,
gniazdo	zmienna lub stała określająca numer gniazda (**0** – **3**).

Opis:
Jeśli system pracuje jako serwer, może być wymagane odczytanie numeru portu połączonego klienta. Można to wykorzystać do logowania tylko wybranych klientów czy jako element systemu zabezpieczeń.

Przykład:

```
Dim W As Word
W = GetdstPort(i)        ' pobierz numer portu dla gniazda o numerze i
```

GETSOCKET()

Przeznaczenie:
Otwiera gniazdo do komunikacji protokołem TCP/IP.

Składnia:
rezultat = **GETSOCKET (** *gniazdo* **,** *tryb* **,** *port* **,** *opcje* **)**

gdzie:

rezultat	zwraca numer otwartego gniazda lub 255 jeśli operacja otwarcia się nie powiodła,
gniazdo	numer gniazda **0** - **3** jakie ma być otwarte,
tryb	określa tryb pracy gniazda,
port	określa port jaki używać będzie gniazdo,
opcje	określa dodatkowe opcje.

Opis:
Funkcja próbuje otworzyć uprzednio zamknięte gniazdo. Jeśli operacja się powiedzie funkcja zwraca numer otwartego właśnie gniazda. Niepowodzenie operacji otwarcia sygnalizowane jest zwróceniem wartości 255 (–1 w notacji U2).

Otwarcie gniazda może się odbyć w kilku predefiniowanych trybach: `sock_stream` (1), `sock_dgrm` (2), `sock_ipl_raw` (3) or `macl_raw` (4). Najlepiej zdefiniować numery trybów jako stałe.
Dla komunikacji za pomocą UDP należy podać `sock_dgrm` lub 2.

Następnym parametrem jaki trzeba określić jest port przez jaki odbywać się będzie komunikacja. Można podać dowolną wartość lecz należy pamiętać by każde gniazdo posiadało własny numer. Można także podać 0, wtedy numer portu będzie pobrany ze zmiennej LOCAL_PORT której wartość przypisywana jest podczas konfiguracji instrukcją **CONFIG TCPIP**. Po wykonaniu konfiguracji zwartość zmiennej LOCAL_PORT jesz zwiększana o 1, więc jako parametr *port* można podać 0.

Ostatnim parametrem można określić dodatkowe opcje:
- 0 : brak dodatkowych opcji,
- 128 : nadaje/odbiera ogólny komunikat za pomocą UDP,
- 64 : zastosuj wartość z rejestru ze wskazanym czasem przeterminowania,
- 32 : jeśli nie używa nie opóźnionego ACK,
- 16 : jeśli nie używa „syndromu nienormalnego okna”.

Dodatkowych informacji należy szukać w nocie katalogowej układu W3100A.

Po poprawnej inicjalizacji i otwarciu gniazda można użyć funkcji **SOCKETCONNECT** by połączyć się z klientem sieci lub **SOCKETLISTEN** by rozpocząć nasłuchiwanie (praca jako serwer).

Przykład:

```
I = Getsocket(0 , Sock_stream , 5000 , 0)     'przydziel nowe gniazdo
```

SOCKETCONNECT()

Przeznaczenie:
Nawiązuje połączenie z serwerem TCP/IP.

Składnia:
rezultat = **SOCKETCONNECT(** *gniazdo* **,** *adres_IP* **,** *port* **)**

gdzie:

rezultat	zmienna typu Byte gdzie funkcja zwraca 0 jeśli połączenie zostanie nawiązane, lub 1 gdy wystąpił błąd,
gniazdo	numer z zakresu **0** – **3** określający numer gniazda,
adres_IP	adres IP serwera z którym trzeba się połączyć,
port	parametr określający numer portu.

Opis:
Funkcja przeznaczona jest tylko do nawiązania połączenia z serwerem. Należy przy tym podać adres IP oraz portu, gdyż standardowo serwery posiadają dedykowane numery portów. Na przykład serwer HTTP (web server) używa portu 80.
Adres IP można podać w postaci zwykłego zapisu (4 liczby oddzielone kropkami) lub jako zmienną typu Long, w której bajt MSB określa część LSB numeru IP.

Po nawiązaniu połączenia serwer może odpowiedzieć ciągiem danych. Dane te zależą od używanego protokołu. Większość serwerów wysyła tekst powitalny, zwany banerem.

Można wysyłać lub odebrać dane po nawiązaniu połączenia. Serwer może także zakończyć połączenie lub też można zakończyć połączenie samodzielnie. To także zależy od używanego protokołu.

Przykład:

```
J = SocketConnect(i , 194.109.6.52 , 25)        ' serwer SMTP
```

SOCKETLISTEN

Przeznaczenie:
Otwiera gniazdo w trybie serwera (nasłuch).

Składnia:
SOCKETLISTEN *numer*

gdzie:
numer numer z zakresu **0** – **3** określający numer gniazda.

Opis:
Wykonanie tej instrukcji powoduje, że wybrane gniazdo będzie nasłuchiwać portu, wybranego wcześniej za pomocą funkcji **GETSOCKET**. Jednocześnie można nasłuchiwać maksimum 4 gniazda.

Po zamknięciu połączenia – przez klienta lub przez serwer – ustanowione musi być nowe połączenie by móc ponownie użyć instrukcji **SOCKETLISTEN**.

Przykład:

```
I = Getsocket(0 , Sock_stream , 5000 , 0)        ' otwórz nowe gniazdo
Socketlisten I                                   ' nasłuchuj
#if Debug
  Print "Nasłuchuję gniazda: " ; I
#endif
```

SOCKETSTAT()

Przeznaczenie:
Zwraca informacje o wybranym gnieździe.

Składnia:
rezultat **= SOCKETSTAT(** *gniazdo* **,** *tryb* **)**

gdzie:
rezultat zmienna typu Word gdzie funkcja zwróci stan gniazda,
gniazdo numer z zakresu **0** – **3** określający numer gniazda.
tryb parametr określający jaką informację ma zwrócić funkcja.

Opis:
Funkcja **SOCKETSTAT** aktualnie zwraca 3 typy informacji. Jedną z nich wybiera się podając odpowiedni parametr (*tryb*):

SEL_CONTROL (0) Funkcja zwraca wartość z rejestru statusu;

SEL_SEND (1) Funkcja zwraca liczbę bajtów które można umieścić w buforze transmisji;

SEL_RECV (2) Funkcja zwraca liczbę bajtów które są umieszczone w buforze odbiorczym;

Jeśli wybrany zostanie tryb **0** (**SEL_CONTROL**) funkcja zwróci jedną z poniższych wartości:

Wartość	*Stan*	*Opis*
0	SOCK_CLOSED	Połączenie zamknięte
1	SOCK_ARP	Oczekiwanie na odpowiedź po nadaniu ARP
2	SOCK_LISTEN	Oczekiwanie na ustawienie połączenia przez klienta podczas pracy w trybie pasywnym
3	SOCK_SYSSENT	Oczekiwanie na SYN, ACK po transmisji SYN dla ustawienia połączenia podczas pracy w trybie aktywnym
4	SOCK_SYSSENT_ACK	Zakończenie ustawienia połączenia po odebraniu SYN, ACK i nadanie ACK podczas pracy w trybie aktywnym
5	SOCK_SYNCRECV	SYN, ACK został wysłany po odebraniu SYN przez klienta pracującego w trybie nasłuchu, tryb pasywny
6	SOCK_ESTABLISHED	Ustawienie połączenia zostało zakończone w trybie aktywnym/pasywnym
7	SOCK_CLOSE_WAIT	Połączenie zostało przerwane
8	SOCK_LAST_ACK	Połączenie zostało przerwane
9	SOCK_FIN_WAIT1	Połączenie zostało przerwane
10	SOCK_FIN_WAIT2	Połączenie zostało przerwane
11	SOCK_CLOSING	Połączenie zostało przerwane
12	SOCK_TIME_WAIT	Połączenie zostało przerwane
13	SOCK_RESET	Połączenie zostało przerwane po odebraniu pakietu zerującego
14	SOCK_INIT	Gniazdo jest inicjalizowane
15	SOCK_UDP	Właściwy kanał jest zainicjalizowany w trybie UDP
16	SOCK_RAW	Właściwy kanał jest zainicjalizowany w trybie warstwy RAW
17	SOCK_UDP_ARP	Oczekiwanie na odpowiedź po przesłaniu pakietu ARP do miejsca docelowego dla transmisji UDP
18	SOCK_UDP_DATA	Trwa transmisja w trybie UDP RAW
19	SOCK_RAW_INIT	Układ W3100A zainicjalizowany w trybie MAC warstwy RAW

Przykład:

```
Tempw = SocketStat(i , 0)           ' pobierz słowo statusu
Select Case Tempw
  Case Sock_established
End Select
```

TCPREAD()

Przeznaczenie:
Odczytuje dane z otwartego gniazda.

Składnia:
rezultat = **TCPREAD(** *gniazdo* **,** *zmienna* [**,** *bajtów*] **)**

gdzie:

rezultat	zmienna typu Word gdzie funkcja umieści liczbę aktualnie odebranych bajtów z gniazda,
gniazdo	numer gniazda (**0** – **3**) za pomocą którego funkcja odbiera dane,
zmienna	nazwa zmiennej gdzie funkcja ma umieścić odebrane dane,
bajtów	ilość bajtów jak ma zostać odczytana. Ważna tylko dla zmiennych nie będących ciągami znaków.

Opis:
W przypadku gdzie jako parametr *zmienna* funkcji **TCPREAD** użyto zmiennej typu String procedura będzie czekać aż odebrany będzie znak końca linii (CR+LF). Nie ma zatem potrzeby podawania ilości odbieranych bajtów.

Uwaga! Należy pamiętać iż odebrany ciąg znaków będzie pozbawiony CR+LF.

Dla innych typów danych należy podać ile bajtów ma być odebranych. Nie jest jednak sprawdzane czy podana ilość bajtów jest większa niż rozmiar zmiennej. Dlatego odebranie 2 bajtów dla zmiennej typu Byte spowoduje nadpisanie następnego bajtu.

Dla zmiennych typu String funkcja nie powoduje nadpisania dalszych bajtów poza ciągiem. Przykładowo: ciąg znaków ma długość 10 bajtów a odbierany posiada ich 80 – odebrane zostanie tylko 10 znaków, a funkcja zakończy swoje działanie po odebraniu CR+LF.

Uwaga! Jeśli w buforze odbiorczym nie ma wystarczającej ilości wolnych bajtów funkcja zaczeka aż w buforze zwolni się miejsce lub gniazdo zostanie zamknięte.

Przykład:
Zobacz przykład dla funkcji **TCPWRITESTR**

TCPWRITE()

Przeznaczenie:
Przesyła dane za pomocą otwartego gniazda.

Składnia:
rezultat = **TCPWRITE(** *gniazdo* **,** *zmienna* [**,** *bajtów*] **)**

rezultat = **TCPWRITE(** *gniazdo* **, EEPROM ,** *adres* **,** *bajtów* **)**

gdzie:

rezultat	zmienna typu Word gdzie funkcja umieści liczbę aktualnie wysłanych bajtów. Jeśli w bufor jest pełen zwraca 0,
gniazdo	numer gniazda (**0** – **3**) za pomocą którego funkcja ma przesłać dane,

zmienna	nazwa zmiennej lub stała która ma zostać wysłana,
bajtów	ilość wysyłanych bajtów. Ważna tylko dla zmiennych nie będących ciągami znaków i dla danych z pamięci EEPROM,
adres	adres w pamięci EEPROM gdzie umieszczono pierwszy wysyłany bajt.

Opis:
Pierwsza postać funkcji pozwala na wysłanie zawartości dowolnej zmiennej (oprócz zmiennych bitowych). Dla typu String nie podaje się ilości bajtów gdyż funkcja sama określa długość ciągu. Dla innych typów ilość bajtów należy obowiązkowo podać. Funkcja może także przesyłać zawartość tablic, wtedy należy podać także indeks wysyłanego elementu. Przykładowo: `zmienna(2)`.

Uwaga! Dla ciągów znaków funkcja nie dodaje znaków CR+LF.

Druga postać instrukcji pozwala na przesyłanie zawartości wewnętrznej pamięci EEPROM. Należy tylko podać adres pierwszego bajtu i ilość przesyłanych bajtów.

Jeśli w buforze nadawczym jest wystarczająca ilość wolnych bajtów funkcja zwraca wartość taką samą jak została podana.

Przykład:

```
Tempw = Tcpwrite(i , "HTTP/1.0 200 OK{013}{010}")
```

TCPWRITESTR()

Przeznaczenie:
Przesyła dane tekstowe za pomocą otwartego gniazda.

Składnia:
rezultat = **TCPWRITESTR(** *gniazdo* **,** *zmienna* **,** *opcje* **)**

gdzie:

rezultat	zmienna typu Word gdzie funkcja umieści liczbę aktualnie wysłanych bajtów. Jeśli w bufor jest pełen zwraca 0,
gniazdo	numer gniazda (**0** – **3**) za pomocą którego funkcja ma przesłać dane,
zmienna	nazwa zmiennej typu String która ma zostać wysłana,
opcje	można podać **0** jeśli wysłana ma być zawartość zmiennej bez znaku końca linii; lub **255** by funkcja sama dodała CR+LF.

Opis:
Funkcja jest specjalnym wariantem poprzedniej funkcji **TCPWRITE**. Przeznaczona jest przede wszystkim do wysyłania ciągów znaków. W zależności od ostatniego parametru funkcja dodaje znaki CR+LF.

Przykład:

```
Tempw = Tcpwritestr(i , "MAIL FROM:<tcpip@mcselec.com>{013}{010}", 0) '
wyślij adres nadawcy
Tempw = Tcpread(i , S)                                                 '
odbierz odpowiedź
```

UDPREAD()

Przeznaczenie:
Odczytuje dane za pomocą protokołu UDP.

Składnia:
rezultat = **UDPREAD(** *gniazdo* **,** *zmienna* [**,** *bajtów*] **)**

gdzie:

rezultat	zmienna typu Word gdzie funkcja umieści liczbę aktualnie odebranych bajtów z gniazda,
gniazdo	numer gniazda (**0** – **3**) za pomocą którego funkcja odbiera dane,
zmienna	nazwa zmiennej gdzie funkcja ma umieścić odebrane dane,
bajtów	ilość bajtów jak ma zostać odczytana. Ważna tylko dla zmiennych nie będących ciągami znaków.

Opis:
W przypadku gdzie jako parametr *zmienna* funkcji **UDPREAD** użyto zmiennej typu String procedura będzie czekać aż odebrany będzie znak końca linii (CR+LF). Nie ma zatem potrzeby podawania ilości odbieranych bajtów.

Uwaga! Należy pamiętać iż odebrany ciąg znaków będzie pozbawiony CR+LF.

Dla innych typów danych należy podać ile bajtów ma być odebranych. Nie jest jednak sprawdzane czy podana ilość bajtów jest większa niż rozmiar zmiennej. Dlatego odebranie 2 bajtów dla zmiennej typu Byte spowoduje nadpisanie następnego bajtu.

Dla zmiennych typu String funkcja nie powoduje nadpisania dalszych bajtów poza ciągiem. Przykładowo: ciąg znaków ma długość 10 bajtów a odbierany posiada ich 80 – odebrane zostanie tylko 10 znaków, a funkcja zakończy swoje działanie po odebraniu CR+LF.

Uwaga! Jeśli w buforze odbiorczym nie ma wystarczającej ilości wolnych bajtów funkcja zaczeka aż w buforze zwolni się miejsce lub gniazdo zostanie zamknięte.

Funkcja **SOCKETSTAT** w przypadku protokołu UDP zwraca zawsze wartość o 8 bajtów większą. Spowodowane jest to tym, że protokół UDP zawsze dodaje nagłówek, który zawiera m.in. długość pakietu danych, adres IP i numer portu równorzędnego komputera (połączenie jest typu peer-to-peer).
Funkcja UDPREAD rozkodowywuje nagłówek i poszczególne jego składowe umieszcza w zmiennych: `PeerSize`, `PeerAdress`, `PeerPort`.

Uwaga! Powyższe zmienne należy wcześniej zadeklarować w programie dokładnie w ten sposób:

```
Dim PeerSize As Integer , PeerAdress As Long , PeerPort As Word
```

Przykład:

```
Result = Socketstat(idx , Sel_recv)      'pobierz liczbę oczekujących bajtów
   If Result > 0 Then
      Print "Ilość oczekujących bajtów : " ; Result
      Temp2 = Result - 8                 '8 pierwszych bajtów zawsze zawiera
nagłówek UDP
                                         'zawierający długość, adres IP i
numer portu
      Temp = Udpread(idx , S(1) , Result)       ' odczytaj rezultat
      For Temp = 1 To Temp2
          Print S(temp) ; " " ;                 ' i wydrukuj
      Next
    End If
```

UDPWRITE()

Przeznaczenie:
Przesyła dane za pomocą gniazda protokołem UDP.

Składnia:
rezultat = **UDPWRITE(** *IP* **,** *port* **,** *gniazdo* **,** *zmienna* [**,** *bajtów*] **)**

rezultat = **UDPWRITE(** *IP* **,** *port* **,** *gniazdo* **, EEPROM ,** *adres* **,** *bajtów* **)**

gdzie:

rezultat	zmienna typu Word gdzie funkcja umieści liczbę aktualnie wysłanych bajtów. Jeśli w bufor jest pełen zwraca 0,
IP	adres IP gdzie dane mają być przesłane. Można podać w postaci standardowej (4 wartości oddzielone kropkami) lub jako zmienną typu Long, gdzie część MSB zmiennej zawiera część LSB adresu IP,
port	numer portu do którego przesłane mają być dane,
gniazdo	numer gniazda (**0** – **3**) za pomocą którego funkcja ma przesłać dane,
zmienna	nazwa zmiennej lub stała która ma zostać wysłana,
bajtów	ilość wysyłanych bajtów. Ważna tylko dla zmiennych nie będących ciągami znaków i dla danych z pamięci EEPROM,
adres	adres w pamięci EEPROM gdzie umieszczono pierwszy wysyłany bajt.

Opis:
Pierwsza postać funkcji pozwala na wysłanie zawartości dowolnej zmiennej (oprócz zmiennych bitowych). Dla typu String nie podaje się ilości bajtów gdyż funkcja sama określa długość ciągu. Dla innych typów ilość bajtów należy obowiązkowo podać. Funkcja może także przesyłać zawartość tablic, wtedy należy podać także indeks wysyłanego elementu. Przykładowo: `zmienna(2)`.

Druga postać instrukcji pozwala na przesyłanie zawartości wewnętrznej pamięci EEPROM. Należy tylko podać adres pierwszego bajtu i ilość przesyłanych bajtów.

Jeśli w buforze nadawczym jest wystarczająca ilość wolnych bajtów funkcja zwraca wartość taką samą jak została podana.

Uwaga! Funkcja **UDPWRITE** jest prawie taka sama jak **TCPWRITE**. Ponieważ UPD jest protokołem „bezpołączeniowym", należy podać adres IP i numer portu. UDP wymaga tylko otwartego gniazda.

Przykład:
Zobacz przykład dla funkcji **UDPWRITESTR**

UDPWRITESTR()

Przeznaczenie:
Przesyła dane tekstowe za pomocą protokołu UDP.

Składnia:
rezultat = **UDPWRITESTR(** *IP* **,** *port* **,** *gniazdo* **,** *zmienna* **,** *opcje* **)**

gdzie:

rezultat	zmienna typu Word gdzie funkcja umieści liczbę aktualnie wysłanych bajtów. Jeśli w bufor jest pełen zwraca 0,
gniazdo	numer gniazda (**0** – **3**) za pomocą którego funkcja ma przesłać dane,
IP	adres IP gdzie dane mają być przesłane. Można podać w postaci standardowej (4 wartości oddzielone kropkami) lub jako zmienną typu Long, gdzie część MSB zmiennej zawiera część LSB adresu IP,
port	numer portu do którego przesłane mają być dane,
zmienna	nazwa zmiennej typu String która ma zostać wysłana,
opcje	można podać **0** jeśli wysłana ma być zawartość zmiennej bez znaku końca linii; lub **255** by funkcja sama dodała CR+LF.

Opis:
Funkcja jest specjalnym wariantem poprzedniej funkcji **UDPWRITE**. Przeznaczona jest przede wszystkim do wysyłania ciągów znaków. W zależności od ostatniego parametru funkcja dodaje znaki CR+LF.

Przykład:

```
'-----------------------------------------------------------------------
'                          UDPTEST.BAS
'                   (c) 2002-2003 MCS Electronics
' Uruchom program easytcp.exe po zaprogramowaniu procesora i naciśnij
przycisk UDP
'-----------------------------------------------------------------------
```

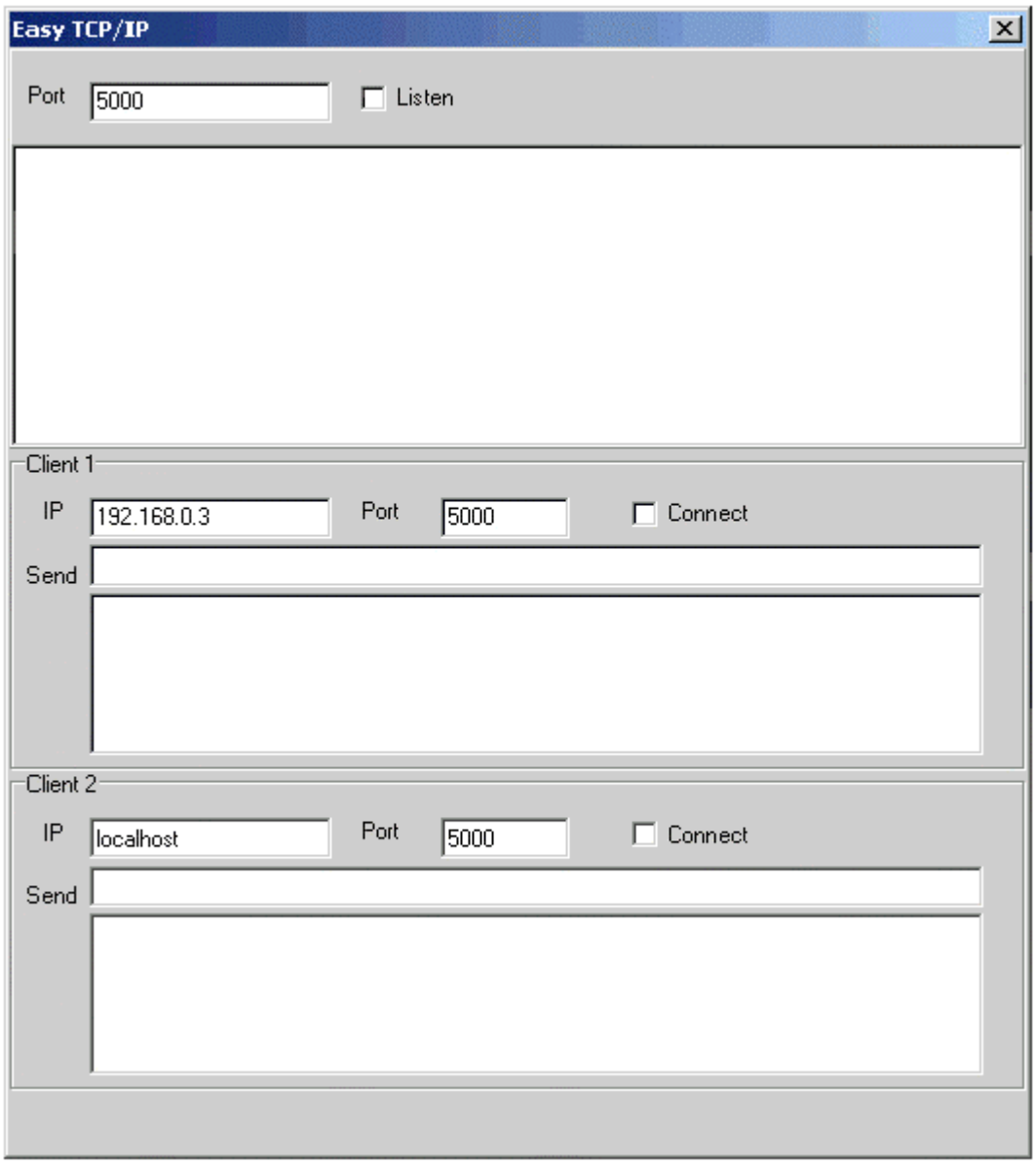
```
$regfile = "m161def.dat"      ' użyty procesor
$crystal = 4000000            ' użyty kwarc
$baud = 19200                 ' szybkość pracy UART

Const Sock_stream = $01                    ' Tcp
Const Sock_dgram = $02                     ' Udp
Const Sock_ipl_raw = $03                   ' Ip Layer Raw Sock
Const Sock_macl_raw = $04                  ' Mac Layer Raw Sock
Const Sel_control = 0                      ' Confirm Socket Status
Const Sel_send = 1                         ' Confirm Tx Free Buffer Size
Const Sel_recv = 2                         ' Confirm Rx Data Size

'Status gniazda
Const Sock_closed = $00         ' Status Of Connection Closed
Const Sock_arp = $01            ' Status Of Arp
Const Sock_listen = $02         ' Status Of Waiting For Tcp Connection Setup
Const Sock_synsent = $03        ' Status Of Setting Up Tcp Connection
Const Sock_synsent_ack = $04    ' Status Of Setting Up Tcp Connection
Const Sock_synrecv = $05        ' Status Of Setting Up Tcp Connection
Const Sock_established = $06    ' Status Of Tcp Connection Established
Const Sock_close_wait = $07     ' Status Of Closing Tcp Connection
Const Sock_last_ack = $08       ' Status Of Closing Tcp Connection
Const Sock_fin_wait1 = $09      ' Status Of Closing Tcp Connection
Const Sock_fin_wait2 = $0a      ' Status Of Closing Tcp Connection
Const Sock_closing = $0b        ' Status Of Closing Tcp Connection
Const Sock_time_wait = $0c      ' Status Of Closing Tcp Connection
Const Sock_reset = $0d          ' Status Of Closing Tcp Connection
Const Sock_init = $0e           ' Status Of Socket Initialization
Const Sock_udp = $0f            ' Status Of Udp
Const Sock_raw = $10            ' Status of IP RAW


$lib "tcpip.lib"                ' używamy biblioteki dodatkowej tcpip.lib
Print "Inicjalizacja, ustaw IP na 192.168.0.8"
Enable Interrupts               ' włącz przerwania !
Config Tcpip = Int0 , Mac = 12. 128.12.34.56.78 , Ip = 192.168.0.8 , Submask
= 255.255.255.0 , Gateway = 0.0.0.0 , Localport = 1000 , Tx = $55 , Rx = $55

'użyj poniższej wersji gdy używana jest brama domyślna
'Config Tcpip = Int0 , Mac = 12. 128.12.34.56.78 , Ip = 192.168.0.8 , Submask
= 255.255.255.0 , Gateway = 192.168.0.1 , Localport = 1000 , Tx = $55 , Rx =
$55

Dim Idx As Byte                          ' numer gniazda
Dim Result As Word                       ' rezultat
Dim S(80) As Byte
Dim Sstr As String * 20
Dim Temp As Byte , Temp2 As Byte         ' bajty tymczasowe
'-------------------------------------------------------------------------
'gdy używany jest UDP należy zdefiniować poniższe zmienne dokładnie tak jak
tutaj!!
Dim Peersize As Integer , Peeraddress As Long , Peerport As Word
'-------------------------------------------------------------------------
Declare Function Ipnum(ip As Long) As String       ' a handy function

'like with TCP, we need to get a socket first
'note that for UDP we specify sock_dgram
Idx = Getsocket(idx , Sock_dgram , 5000 , 0)       ' get socket for UDP mode,
specify port 5000
Print "Socket " ; Idx ; " " ; Idx

'UDP to protokół bezpołączeniowy co oznacza, że nie można nasłuchiwać, łączyć
się lub odczytywać statusu
'Można tylko nadawać lub odbierać dane tak w ten sam sposób jak dla TCP /IP.
'Lecz ponieważ nie ma protokołu łączenia, należy podać adres IP i port
komputera docelowego
'Porównując do funkcji dla TCP/IP można wysyłać dokładnie tak samo, lecz z
dodatkowo określonym adresem IP i numerem portu
Do
```

```
   Temp = Inkey()                          ' czekaj na znak
   If Temp = 27 Then                       ' czy klawisz ESC
      Sstr = "Witaj"
      Result = Udpwritestr(192.168.0.3 , 5000 , Idx , Sstr , 255)
   End If
   Result = Socketstat(idx , Sel_recv)     'pobierz liczbę oczekujących
bajtów
   If Result > 0 Then
      Print "Ilość oczekujących bajtów : " ; Result
      Temp2 = Result - 8                   '8 pierwszych bajtów zawsze zawiera
nagłówek UDP
                                           'zawierający długość, adres IP i
numer portu
      Temp = Udpread(idx , S(1) , Result)       ' odczytaj rezultat
      For Temp = 1 To Temp2
          Print S(temp) ; " " ;                 ' i wydrukuj
      Next
      Print
      Print Peersize ; " " ; Peeraddress ; " " ; Peerport       ' zmienne te
są wypełniane przez UDPREAD
      Print Ipnum(peeraddress)             ' wydrukuj numer IP w przyjazny
sposób
      Result = Udpwrite(192.168.0.3 , Peerport , Idx , S(1) , Temp2)
       ' prześlij odebrane dane z powrotem
    End If
Loop

'Przykład powyższy oczekuje na dane i wysyła je z powrotem; dlatego też temp2
jest zmniejszana o 8, czyli rozmiar bufora

'Funkcja ta może być używana do wyświetlania adresu IP w normalny sposób
Function Ipnum(ip As Long) As String
    Local T As Byte , J As Byte
    Ipnum = ""
    For J = 1 To 4
      T = Ip And 255
      Ipnum = Ipnum + Str(t)
      If J < 4 Then Ipnum = Ipnum + "."
      Shift Ip , Right , 8
    Next
End Function

End
```

Program Easy TCP/IP

Aplikacja `easytcpip.exe` jest przeznaczona do testowania różnorodnych funkcji związanych z płytką testową **Easy TCP/IP.**

Jeśli uruchomisz program `easytcpip.exe` powinno się pojawić następujące okno:

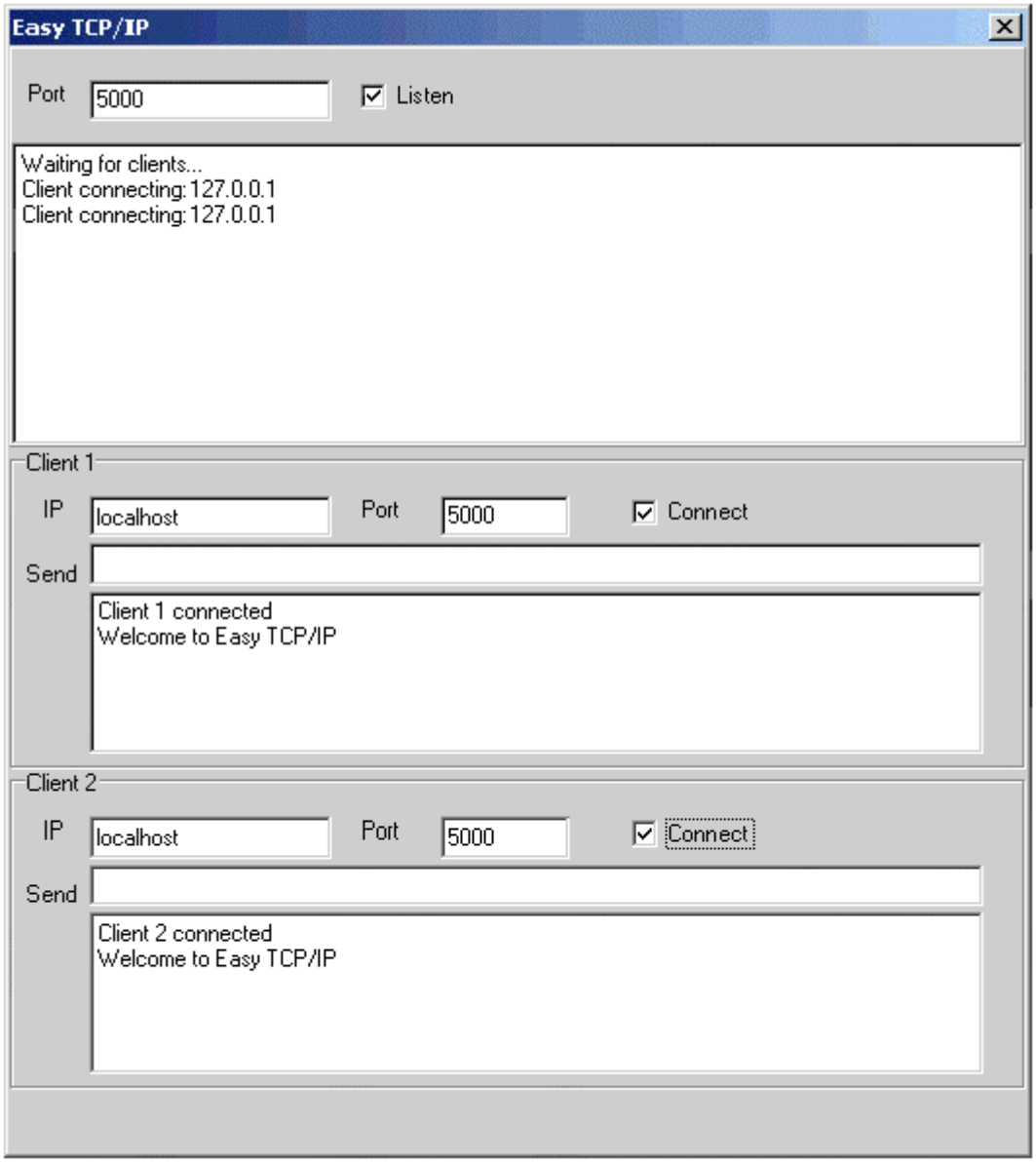


Program może pracować w roli serwera oraz pozwala na przyłączenie się do serwera dwóch klientów.
By przetestować pracę twojej karty sieciowej wykonaj następujące operacje:

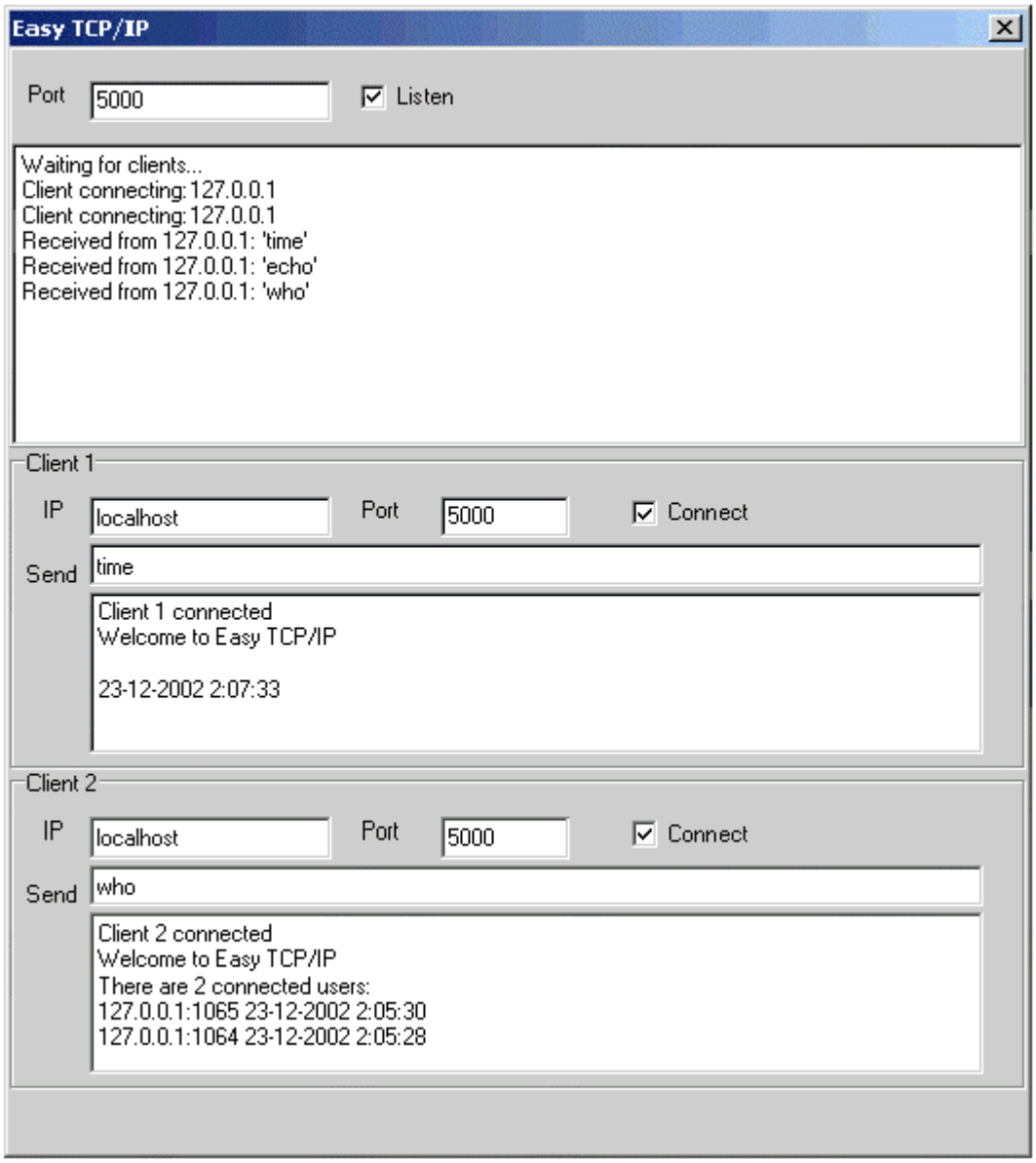
- Wpisz liczbę 5000 w polu **Port** serwera (na samej górze),
- Wpisz taką samą wartość w polach **Port** poszczególnych klientów,
- Zaznacz pole **Listen**,
- Wpisz „localhost" w polu **IP** sekcji 'Client1',
- Wpisz „localhost" w polu **IP** sekcji 'Client2',
- Zaznacz pole **Connect** w sekcji 'Client1',

- Zaznacz pole **Connect** w sekcji 'Client2'

Okno powinno wyglądać mniej więcej tak:



Po uzyskaniu połączenia przez poszczególnych klientów serwer wysyła tekst powitania: 'Welcome to Easy TCP/IP'.
Możesz teraz rozpocząć wysyłanie komend do serwera używając pól **Send**. Wypróbuj polecenie: `TIME` lub `WHO` (Nie zapomnij po wpisaniu polecenia nacisnąć klawisza [Enter]!). Serwer powinien odpowiedzieć wysyłając aktualny czas lub informacje o połączonych klientach:



Klikając pole **Connect** ponownie, przerywasz połączenie. Możesz także wydać polecenie `EXIT` by uzyskać ten sam rezultat.

Localhost to alias adresu IP 127.0.0.1, który jest generowany automatycznie dla każdego komputera. Możesz także podać właściwy adres IP wpisując: 192.168.0.3 (lub inny, ustalony w polu **Adres IP** konfiguracji protokołu TCP/IP komputera).

Jeśli wszystko zadziałało poprawnie, jesteśmy gotowi do przeprowadzenia kilku następnych testów. W tym celu rozłącz bieżących klientów oraz pozostaw serwer w trybie nasłuchu (pole **Listen** zaznaczone). Uruchom środowisko BASCOM oraz otwórz program `clienttest.bas`. Skompiluj go i zaprogramuj mikrokontroler. Teraz otwórz okno terminala by mieć wgląd do komunikatów przesyłanych przez płytkę **Easy TCP/IP** (Nie zapomnij połączyć płytki z komputerem kablem transmisji szeregowej!).

Przykładowy program otwiera 4 gniazda za pomocą których łączy się z serwerem. Jeśli wszystko działa, powinieneś zaobserwować procesy łączenia się klientów.

W oknie terminala musisz zobaczyć tekst powitalny przesłany przez serwer. Naciskając klawisz [ESC], będziesz poproszony o wydanie jakiejkolwiek komendy. Wpisz `TIME` lub `WHO` oraz naciśnij [Enter]. Komenda ta będzie przesłana do serwera a ty zostaniesz poinformowany o odebraniu komendy. Serwer natomiast wyśle znaną już odpowiedź która pokaże się w oknie terminala.

Jeśli będziesz gotowy naciśnij ponownie klawisz [ESC] i wpisz rozkaz `EXIT`. Teraz połączenie z klientem zostanie zakończone. Zobaczysz to zarówno w oknie programu serwera oraz w oknie terminala. Po rozłączeniu, program Easy TPC/IP musi być uruchomiony ponownie, gdyż inaczej możesz nie uzyskać później połączenia.

Teraz wypróbujemy tryb serwera. Otwórz program `servertest.bas`, skompiluj go i zaprogramuj procesor. W programie Easy TCP/IP wpisz w polu **IP** adres 192.168.0.8 z którym to będziemy się łączyć. Podany adres IP to adres układu W3100A jaki został mu przypisany instrukcją **CONFIG TCPIP**.
Ponieważ serwer nasłuchuje portu 5000, także musimy taki sam numer wpisać w polu **Port**. Teraz zaznacz pole **Connect**. Powinieneś zobaczyć tekst jaki został wysłany przez serwer BASCOM.
Możesz wysłać polecenia do serwera wpisując je i zatwierdzając klawiszem [Enter]. Jeśli jesteś gotowy kliknij pole **Connect** ponownie by rozłączyć klienta z serwerem. Wysyłając polecenie `TIME`, serwer powinien zwrócić czas, przykładowo 12:00:00. Wysyłając polecenie `EXIT`, połączenie zostanie przerwane.

Jeśli to zadziała, utwórz dwa połączenia i powtórz test.

Mam nadzieję iż teraz rozumiesz jakie to jest proste. Jest także parę innych programów na które powinieneś zwrócić uwagę:

- `pop3.bas` odczytujący liczbę wiadomości jakie znajdują się w twojej skrzynce pocztowej,
- `smtp.bas` wysyłający prostą wiadomość e-mail na twoje konto,
- `webserver.bas` który wyświetla parę prostych stron www.

Przykłady są bardzo proste i nie ukazują wszystkich możliwości, więcej informacji należy szukać w dokumentach opisujących poszczególne protokoły (do odnalezienia w sieci Internet, słowo kluczowe: RFC). Przykłady dotyczące protokołów POP3 oraz SMTP wymagają połączenia się z siecią Internet. Dlatego też adres domyślnej bramy (gateway) musi być poprawnie określony w opcjach dotyczących karty sieciowej oraz w poleceniu **CONFIG TCPIP**.
Program demonstracyjny `webserver.bas` nie wymaga aktywnego połączenia z siecią Internet, gdyż sam jest serwerem sieci Web.

Inne programy

Jest także dostępnych kilka wygodnych programów zainstalowanych razem z systemem Windows.

`netstat`	Wyświetla listę aktywnych połączeń z twoim komputerem PC.
`nbtstat`	Pokazuje statystyki protokołów i bieżące połączenia w ramach TCP/IP używając NBT(Netbios over TCP/IP)

`ping`	Wysyła pakiet kontrolny pod określony adres IP.
`ipconfig`	Wyświetla konfigurację IP.

Zachęcam do poznania tych kilku programów. Większość posiada pomoc z opisem parametrów wywoływana z linii poleceń.

DODATEK A: Jak działa płytka?

Transformator sieciowy bądź zasilacz jest dołączany do gniazda J4. Mostek prostowniczy wraz kondensatorem filtrującym C8 wytwarzają napięcie stałe, które jest później stabilizowane przez dwa osobne stabilizatory +3,3V i +5V.

Moduł IIM7000 zawiera układ W3100A oraz scalony układ Ethernet PHY. Układ W3100A jest używany w tzw. trybie BUS. Dlatego też zastosowany procesor musi posiadać możliwość zaadresowania zewnętrznej pamięci. Dlatego z rodziny AVR wybrany został układ ATMega161L. Procesor jest programowany za pomocą programatora Sample Electronics który używa portu LPT. Układ MAX232 używany jest jako bufor pomiędzy mikroprocesorem a portem RS-232.

Układ 74HC573 służy do zatrzaskiwania młodszego bajtu adresowego (A0-A7) wystawianego na szynie danych – tzw, magistrala multipleksowana. Układ W3100A zajmuje adresy z zakresu **&H**8000-**&H**FFFF. Większość tego obszaru jest używana na potrzeby wewnętrznych buforów transmisji. Pozostała część jest używana przez rejestry statusowe i kontrolne układu W3100A. Dlatego też linia A15 jest zanegowana i steruje bezpośrednio końcówką /CS (*Chip Select*).

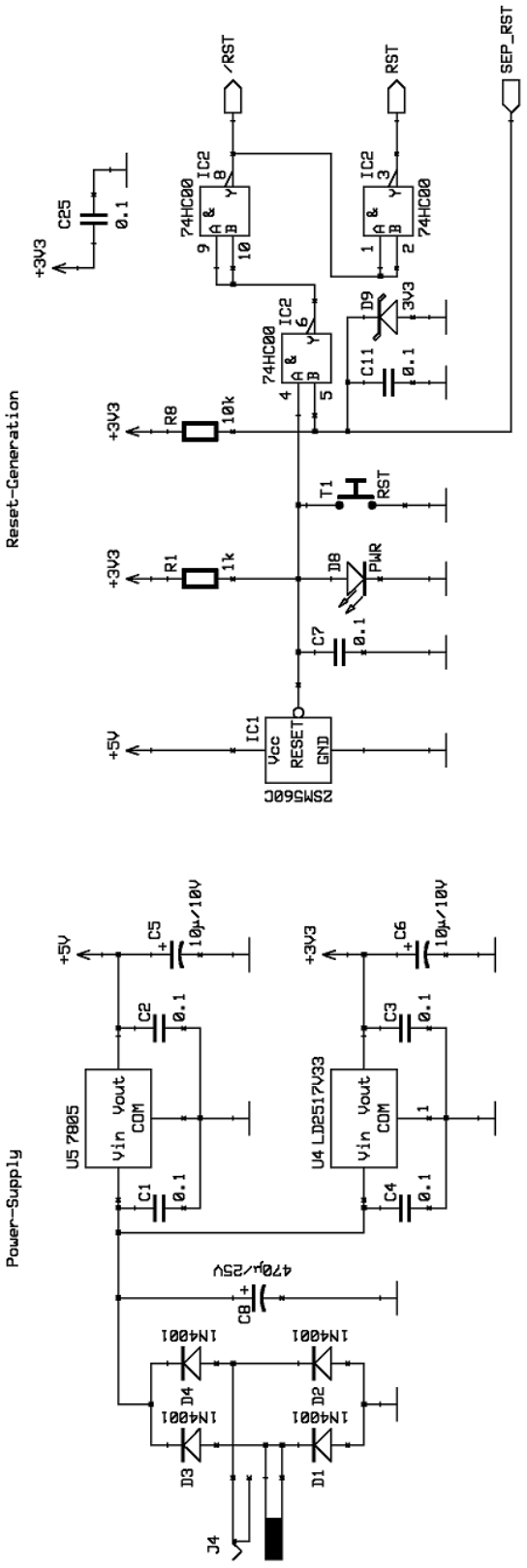
Zworka J5 pozwala na wybranie czy stanem aktywnym sygnału reset ma być stan niski czy niski. Pozwala to na użycie zarówno układu z serii AVR lub też z rodziny 8051.

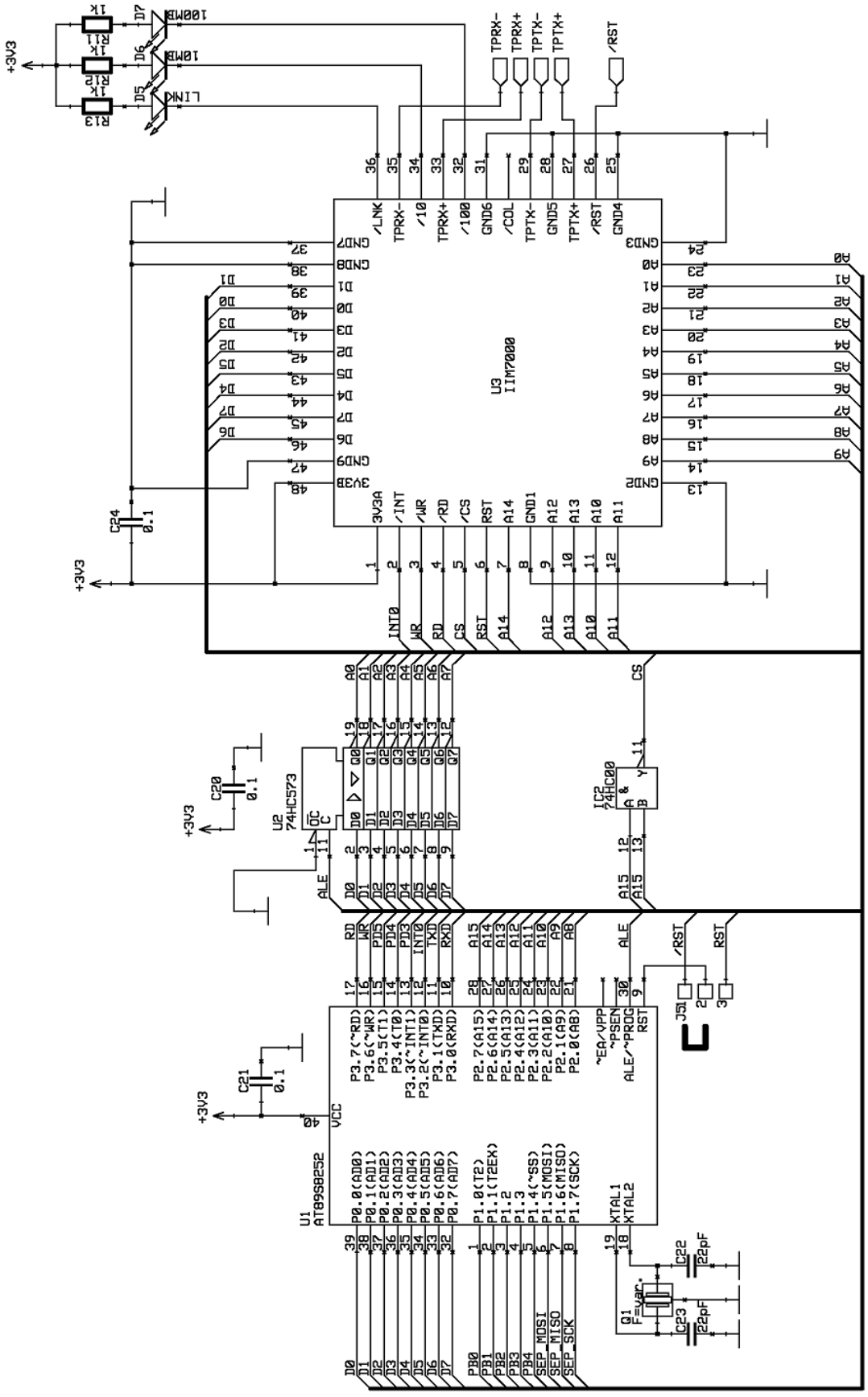
Diody sygnalizacyjne transmisji – Link, 10Mb, 100Mb – są sterowane bezpośrednio przez moduł IIM 7000.

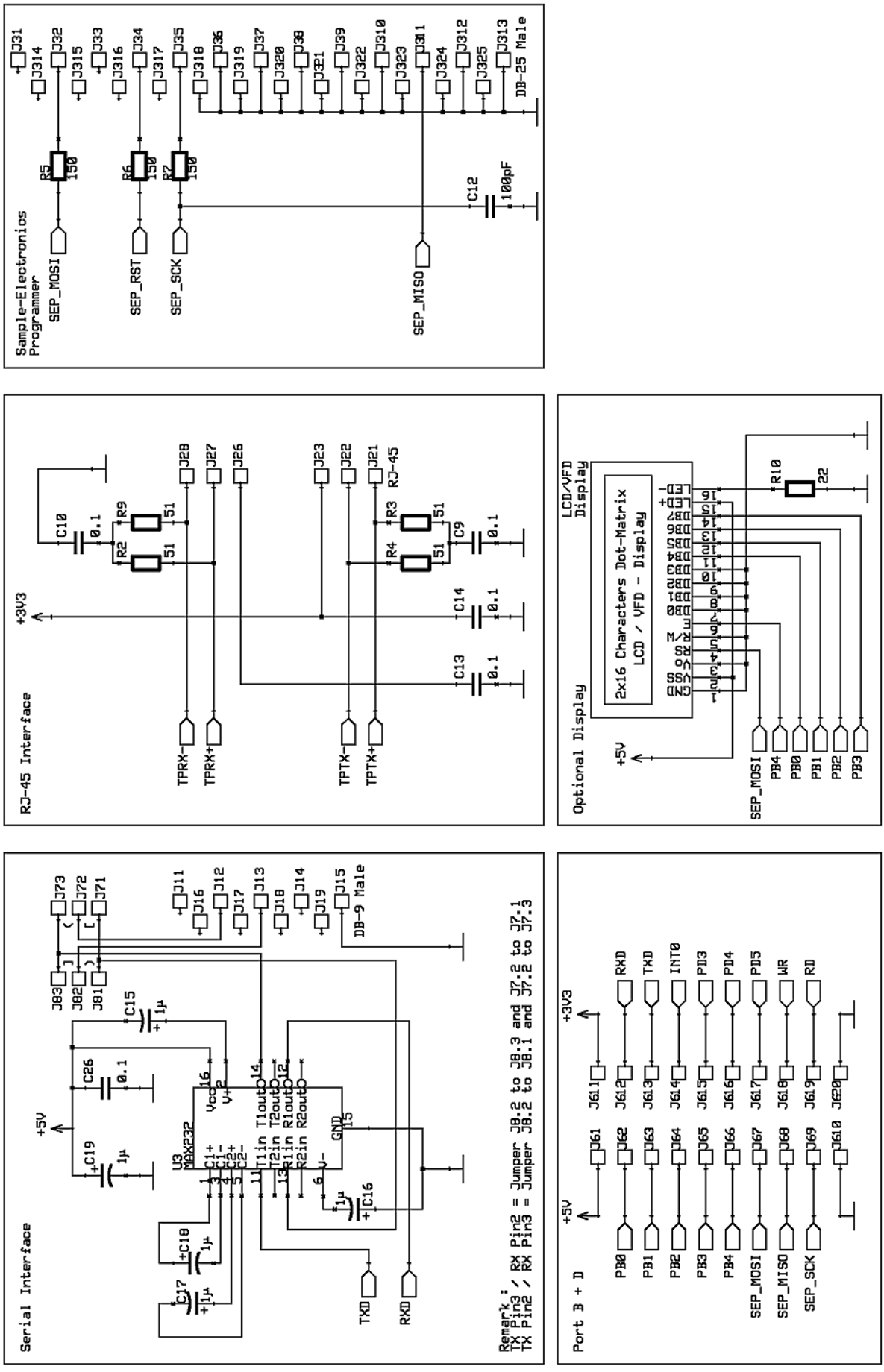
Płytka posiada także obszar prototypowy. W jego pobliżu znajduje się złącze J6 do którego można włożyć/dołączyć dodatkową płytkę. Rozkład sygnałów na tym złączu przedstawia poniższa tabela:

Sygnał	*Końc.*	*Końc.*	*Sygnał*
+5V	**1**	**2**	+3.3V
PORTB.0	**3**	**4**	PORTD.0
PORTB.1	**5**	**6**	PORTD.1
PORTB.2	**7**	**8**	PORTD.2
PORTB.3	**9**	**10**	PORTD.3
PORTB.4	**11**	**12**	PORTD.4
PORTB.5	**13**	**14**	PORTD.5
PORTB.6	**15**	**16**	PORTD.6
PORTB.7	**17**	**18**	PORTD.7
GND	**19**	**20**	GND

DODATEK B: Schematy ideowe.







DODATEK C: Tabela przypisań portów do określonych usług.

Nazwa usługi	*Port/Protokół*	*Komentarz*
tcpmux	1/tcp	# TCP port service multiplexer
echo	7/tcp	
echo	7/udp	
discard	9/tcp	sink null
discard	9/udp	sink null
systat	11/tcp	users
daytime	13/tcp	
daytime	13/udp	
netstat	15/tcp	
qotd	17/tcp	quote
msp	18/tcp	# message send protocol
msp	18/udp	# message send protocol
chargen	19/tcp	ttytst source
chargen	19/udp	ttytst source
ftp-data	20/tcp	
ftp	21/tcp	
fsp	21/udp	fspd
ssh	22/tcp	# SSH Remote Login Protocol
ssh	22/udp	# SSH Remote Login Protocol
telnet	23/tcp	#
	24	private
smtp	25/tcp	
mail	26	unassigned
time	37/tcp	time server
time	37/udp	time server
rlp	39/udp	resource # resource location
nameserver	42/tcp	name # IEN 116
whois	43/tcp	nicname
re-mail-ck	50/tcp	# Remote Mail Checking Protocol
re-mail-ck	50/udp	# Remote Mail Checking Protocol
domain	53/tcp	nameserver # name-domain server
domain	53/udp	nameserver
mtp	57/tcp	# deprecated
bootps	67/tcp	# BOOTP server
bootps	67/udp	
bootpc	68/tcp	# BOOTP client
bootpc	68/udp	
tftp	69/udp	
gopher	70/tcp	# Internet Gopher
gopher	70/udp	
rje	77/tcp	netrjs
finger	79/tcp	
www	80/tcp	http # WorldWideWeb HTTP
www	80/udp	# HyperText Transfer Protocol
link	87/tcp	ttylink
kerberos	88/tcp	kerberos5 krb5 # Kerberos v5
kerberos	88/udp	kerberos5 krb5 # Kerberos v5
supdup	95/tcp	
	100	reserved
hostnames	101/tcp	hostname # usually from sri-nic
iso-tsap	102/tcp	tsap # part of ISODE.
csnet-ns	105/tcp	cso-ns # also used by CSO name server
csnet-ns	105/udp	cso-ns

Nazwa usługi	*Port/Protokół*	*Komentarz*
#3com-tsmux	106/tcp	poppassd
#3com-tsmux	106/udp	poppassd
rtelnet	107/tcp	# Remote Telnet
rtelnet	107/udp	
pop-2	109/tcp	postoffice # POP version 2
pop-2	109/udp	
pop-3	110/tcp	# POP version 3
pop-3	110/udp	
sunrp	111/tcp	portmapper # RPC 4.0 portmapper TCP
sunrpc	111/udp	portmapper # RPC 4.0 portmapper UDP
auth	113/tcp	authentication tap ident
sftp	115/tcp	
uucp-path	117/tcp	
nntp	119/tcp	readnews untp # USENET News Transfer Protocol
ntp	123/tcp	
ntp	123/udp	# Network Time Protocol
netbios-ns	137/tcp	# NETBIOS Name Service
netbios-ns	137/udp	
netbios-dgm	138/tcp	# NETBIOS Datagram Service
netbios-dgm	138/udp	
netbios-ssn	139/tcp	# NETBIOS session service
netbios-ssn	139/udp	
imap2	143/tcp	imap # Interim Mail Access Proto v2
imap2	143/udp	imap
snmp	161/udp	# Simple Net Mgmt Proto
snmp-trap	162/udp	snmptrap # Traps for SNMP
cmip-man	163/tcp	# ISO mgmt over IP (CMOT)
cmip-man	163/udp	
cmip-agent	164/tcp	
cmip-agent	164/udp	
xdmcp	177/tcp	# X Display Mgr. Control Protocol
xdmcp	177/udp	
nextstep	178/tcp	NeXTStep NextStep # NeXTStep window
nextstep	178/udp	NeXTStep NextStep # server
bgp	179/tcp	# Border Gateway Proto.
bgp	179/udp	
prospero	191/tcp	# Cliff Neuman's Prospero
prospero	191/udp	
irc	194/tcp	# Internet Relay Chat
irc	194/udp	
smux	199/tcp	# SNMP Unix Multiplexer
smux	199/udp	
at-rtmp	201/tcp	# AppleTalk routing
at-rtmp	201/udp	
at-nbp	202/tcp	# AppleTalk name binding
at-nbp	202/udp	
at-echo	204/tcp	# AppleTalk echo
at-echo	204/udp	
at-zis	206/tcp	# AppleTalk zone information
at-zis	206/udp	
qmtp	209/tcp	# The Quick Mail Transfer Protocol
qmtp	209/udp	# The Quick Mail Transfer Protocol
z3950	210/tcp	wais # NISO Z39.50 database
z3950	210/udp	wais
ipx	213/tcp	# IPX
ipx	213/udp	
imap3	220/tcp	# Interactive Mail Access
imap3	220/udp	# Protocol v3

Nazwa usługi	*Port/Protokół*	*Komentarz*
rpc2portmap	369/tcp	
rpc2portmap	369/udp	# Coda portmapper
codaauth2	370/tcp	
codaauth2	370/udp	# Coda authentication server
ulistserv	372/tcp	# UNIX Listserv
ulistserv	372/udp	
https	443/tcp	# MCom
https	443/udp	# MCom
snpp	444/tcp	# Simple Network Paging Protocol
snpp	444/udp	# Simple Network Paging Protocol
saft	487/tcp	# Simple Asynchronous File Transfer
saft	487/udp	# Simple Asynchronous File Transfer
npmp-local	610/tcp	dqs313_qmaster # npmp-local / DQS
npmp-local	610/udp	dqs313_qmaster # npmp-local / DQS
npmp-gui	611/tcp	dqs313_execd # npmp-gui / DQS
npmp-gui	611/udp	dqs313_execd # npmp-gui / DQS
hmmp-ind	612/tcp	dqs313_intercell # HMMP Indication / DQS
hmmp-ind	612/udp	dqs313_intercell # HMMP Indication / DQS
Specyficzne usługi systemu UNIX		
exec	512/tcp	
biff	512/udp	comsat
login	513/tcp	
who	513/udp	whod
shell	514/tcp	cmd # no passwords used
syslog	514/udp	
printer	515/tcp	spooler # line printer spooler
talk	517/udp	
ntalk	518/udp	
route	520/udp	router routed # RIP
timed	525/udp	timeserver
tempo	526/tcp	newdate
courier	530/tcp	rpc
conference	531/tcp	chat
netnews	532/tcp	readnews
netwall	533/udp	# -for emergency broadcasts
uucp	540/tcp	uucpd # uucp daemon
afpovertcp	548/tcp	# AFP over TCP
afpovertcp	548/udp	# AFP over TCP
remotefs	556/tcp	rfs_server rfs # Brunhoff remote filesystem
klogin	543/tcp	# Kerberized `rlogin' (v5)
kshell	544/tcp	krcmd# Kerberized `rsh' (v5)
kerberos-adm	749/tcp	# Kerberos `kadmin' (v5)
webster	765/tcp	# Network dictionary
webster	765/udp	
ingreslock	1524/tcp	
ingreslock	1524/udp	
prospero-np.	1525/tcp	# Prospero non-privileged
prospero-np.	1525/udp	
datametrics	1645/tcp	old-radius # datametrics / old radius entry
datametrics	1645/udp	old-radius # datametrics / old radius entry
sa-msg-port	1646/tcp	old-radacct # sa-msg-port / old radacct entry
sa-msg-port	1646/udp	old-radacct # sa-msg-port / old radacct entry
radius	1812/tcp	# Radius
radius	1812/udp	# Radius
radacct	1813/tcp	# Radius Accounting
radacct	1813/udp	# Radius Accounting
cvspserver	2401/tcp	# CVS client/server operations

Nazwa usługi	*Port/Protokół*	*Komentarz*
cvspserver	2401/udp	# CVS client/server operations
venus	2430/tcp	# codacon port
venus	2430/udp	# Venus callback/wbc interface
venus-se	2431/tcp	# tcp side effects
venus-se	2431/udp	#udp sftp side effect
codasrv	2432/tcp	# not used
codasrv	2432/udp	# server port
codasrv-se	2433/tcp	# tcp side effects
codasrv-se	2433/udp	# udp sftp side effect
mysql	3306/tcp	# MySQL
mysql	3306/udp	# MySQL
rfe	5002/tcp	# Radio Free Ethernet
rfe	5002/udp	# Actually uses UDP only
cfengine	5308/tcp	# CFengine
cfengine	5308/udp	# CFengine
bbs	7000/tcp	# BBS service
kerberos4	750/udp	kerberos-iv kdc # Kerberos (server) udp
kerberos4	750/tcp	kerberos-iv kdc # Kerberos (server) tcp
kerberos_master	751/udp	# Kerberos authentication
kerberos_master	751/tcp	# Kerberos authentication
passwd_server	752/udp	# Kerberos passwd server
krb_prop	754/tcp	# Kerberos slave propagation
krbupdate	760/tcp	kreg # Kerberos registration
kpasswd	761/tcp	kpwd # Kerberos "passwd"
kpop	1109/tcp	# Pop with Kerberos
knetd	2053/tcp	# Kerberos de-multiplexor
zephyr-srv	2102/udp	# Zephyr server
zephyr-clt	2103/udp	# Zephyr serv-hm connection
zephyr-hm	2104/udp	# Zephyr hostmanager
eklogin	2105/tcp	# Kerberos encrypted rlogin
Usługi nieoficjalne lecz wymagane dla NetBSD		
supfilesrv	871/tcp	# SUP server
supfiledbg	1127/tcp	# SUP debugging
Usługi Datagram Delivery Protocol		
rtmp	1/ddp	# Routing Table Maintenance Protocol
nbp	2/ddp	# Name Binding Protocol
echo	4/ddp	# AppleTalk Echo Protocol
zip	6/ddp	# Zone Information Protocol
Usługi dodane w dystrybucji Debian GNU/Linux		
poppassd	106/tcp	# Eudora
poppassd	106/udp	# Eudora
mailq	174/tcp	# Mailer transport queue for Zmailer
mailq	174/tcp	# Mailer transport queue for Zmailer
ssmtp	465/tcp	# SMTP over SSL
gdomap	538/tcp	# GNUstep distributed objects
gdomap	538/udp	# GNUstep distributed objects
snews	563/tcp	# NNTP over SSL
ssl-ldap	636/tcp	# LDAP over SSL
omirr	808/tcp	omirrd # online mirror
omirr	808/udp	omirrd # online mirror
rsync	873/tcp	# rsync
rsync	873/udp	# rsync
simap	993/tcp	# IMAP over SSL
spop3	995/tcp	# POP-3 over SSL

Nazwa usługi	*Port/Protokół*	*Komentarz*
socks	1080/tcp	# socks proxy server
socks	1080/udp	# socks proxy server
rmtcfg	1236/tcp	# Gracilis Packeten remote config server
xtel	1313/tcp	# french minitel
support	1529/tcp	# GNATS
cfinger	2003/tcp	# GNU Finger
ninstall	2150/tcp	# ninstall service
ninstall	2150/udp	# ninstall service
afbackup	2988/tcp	# Afbackup system
afbackup	2988/udp	# Afbackup system
icp	3130/tcp	# Internet Cache Protocol (Squid)
icp	3130/udp	# Internet Cache Protocol (Squid)
postgres	5432/tcp	# POSTGRES
postgres	5432/udp	# POSTGRES
fax	4557/tcp	# FAX transmission service (old)
hylafax	4559/tcp	# HylaFAX client-server protocol (new)
noclog	5354/tcp	# noclogd with TCP (nocol)
noclog	5354/udp	# noclogd with UDP (nocol)
hostmon	5355/tcp	# hostmon uses TCP (nocol)
hostmon	5355/udp	# hostmon uses TCP (nocol)
pcANYWHERE32	5631/tcp	
pcANYWHERE32	5631/tcp	
ircd	6667/tcp	# Internet Relay Chat
ircd	6667/udp	# Internet Relay Chat
webcache	8080/tcp	# WWW caching service
webcache	8080/udp	# WWW caching service
tproxy	8081/tcp	# Transparent Proxy
tproxy	8081/udp	# Transparent Proxy
mandelspawn	9359/udp	mandelbrot # network mandelbrot
amanda	10080/udp	# amanda backup services
kamanda	10081/tcp	# amanda backup services(Kerberos)
kamanda	10081/udp	# amanda backup services(Kerberos)
amandaidx	10082/tcp	# amanda backup services
amidxtape	10083/tcp	# amanda backup services
isdnlog	20011/tcp	# isdn logging system
isdnlog	20011/udp	# isdn logging system
vboxd	20012/tcp	# voice box system
vboxd	20012/udp	# voice box system
binkp	24554/tcp	# Binkley
binkp	24554/udp	# Binkley
asp	27374/tcp	# Address Search Protocol
asp	27374/udp	# Address Search Protocol
tfido	60177/tcp	# Ifmail
tfido	60177/udp	# Ifmail
fido	60179/tcp	# Ifmail
fido	60179/udp	# Ifmail