Konwerter Ethernet <–> RS232

AVT–5110

PROJEKT Z OKŁADKI

Konwerter przedstawiony w artykule stanowi w przekonaniu autora najkrótszą i najpewniejszą drogę do skutecznej implementacji interfejsu ethernetowego w nowo projektowanych urządzeniach. Jest on zbudowany w oparciu o znany czytelnikom moduł firmy Tibbo – EM202, który jest w zasadzie sam w sobie konwerterem RS232 <–> Ethernet, jednak od hasła marketingowego do praktycznego zastosowania droga nie zawsze jest prosta.

Rekomendacje:

projekt dedykujemy wszystkim konstruktorom nie posiadającym doświadczenia, a nawet wiedzy na temat działania sieci Ethernet, którzy znaleźli się w sytuacji wymagającej dołączenia projektowanego przez siebie urządzenia do takowej sieci.

PODSTAWOWE PARAMETRY

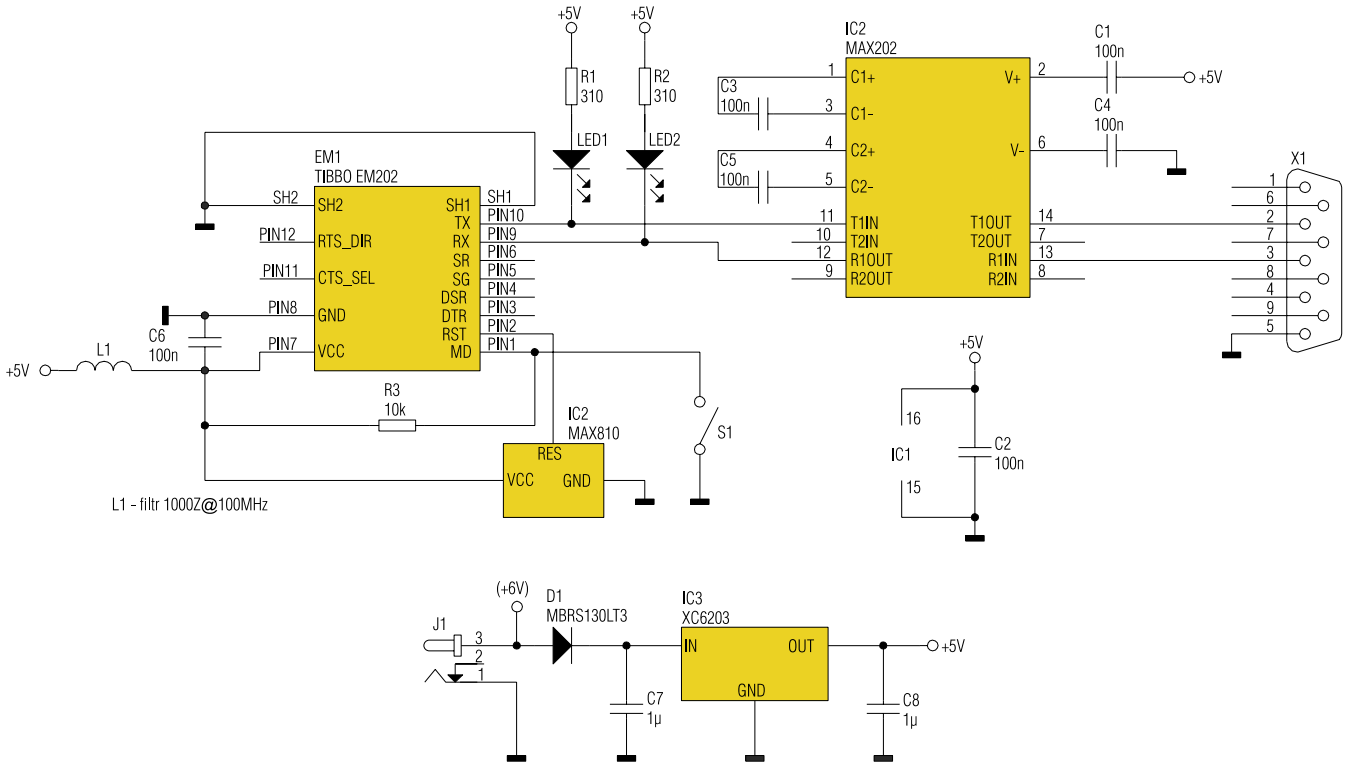
- Płytka o wymiarach 73x45 mm
- Zasilanie 6...8 V (300 mA) może być niestabilizowane
- Praca bez klienta DHCP
- Protokół transportowy TCP
- Współpraca z transceiverem RS232
- Automatyczny wybór trybu pracy portu szeregowego z wyłączoną kontrolą przepływu

Moduły ethernetowe firmy Tibbo z założenia zostały opracowane, aby maksymalnie uprościć obsługę sieci ethernetowej w niemal dowolnym systemie mikroprocesorowym. W praktyce mogą się jednak pojawić pewne pytania i wątpliwości, które postaram się wyjaśnić. Przedstawię krok po kroku, jak moduł taki powinien zostać użyty, by przyniósł maksimum korzyści, zabierając minimum czasu podczas projektowania urządzenia.

Schemat ideowy konwertera przedstawiono na **rys. 1**. Moduł EM202 posiada 12 pinów, spo-

Tab. 1. Zestawienie wyprowadzeń modułu EM202

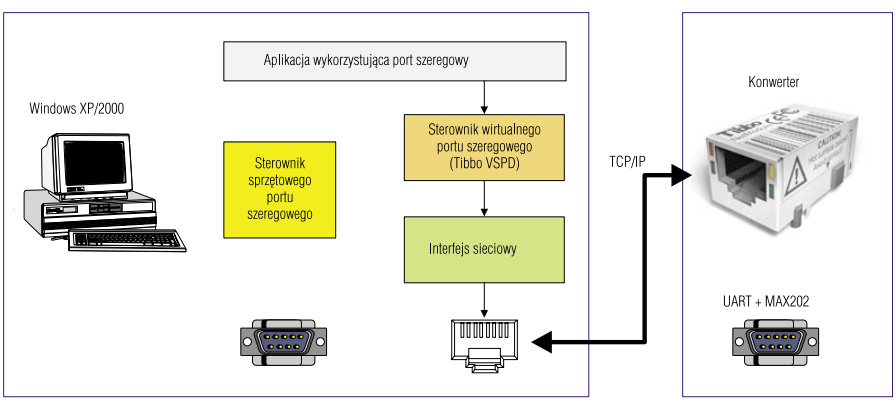
Nr	Pin	Kierunek	Opis
1	MD	We	Pin wyboru trybu pracy. Przy wykorzystaniu standardowego oprogramowania modułu stan panujący na tym wejściu jest sprawdzany na etapie inicjalizacji, po włączeniu zasilania. Kiedy zostanie wykryty stan niski, wówczas modułu przechodzi do trybu konfiguracyjnego wykorzystującego UART. W przeciwnym wypadku moduł rozpoczyna normalną pracę.
2	RST	We	Reset. Po włączeniu zasilania powinien panować na nim stan wysoki do momentu, pojawiania się na wyprowadzeniu VCC napięcia większego niż 4,6 V, po tym fakcie sygnał zerowania powinien utrzymać się jeszcze, przez co najmniej 50 ms. Dokumentacja modułu jasno mówi o unikaniu klasycznych układów RC w obwodzie zerowania. W zamian należy stosować wyspecjalizowane układy zerowania, takie jak np. MAX810, czy DS1812.
3	DTR	Wy	Data Terminal Ready. Poziom logiczny tego wyprowadzenia informuje nas o tym, czy zostało nawiązane połączenie sieciowe pomiędzy zdalnym hostem a modułem. Stanowi informację, iż należy spodziewać się transmisji na linii TX modułu.
4	DSR	We	Data Set Ready. Wejście to może zostać skonfigurowane w ten sposób, iż wymuszony na nim stan będzie stanowił informację dla modułu, aby ten nawiązał połączenie z hostem w sieci.
5	SG	Wy	Green Status Led. Jest to wyjście sterujące zieloną diodą statusową umieszczoną przy złączu RJ45.
6	SR	Wy	Red Status Led. Jest to wyjście sterujące czerwoną diodą statusową umieszczoną przy złączu RJ45.
7	VCC	PWR	Pin zasilania +5 V. Pobór prądu modułu wynosi około 230 mA przy aktywnym linku ethernetowym.
8	GND	PWR	Pin masy.
9	RX	We	UART. Wejście danych, poziomy TTL (5 V).
10	TX	Wy	UART. Wyjście danych, poziomy TTL (5 V).
11	CTS/SEL	We	Clear To Send. Sygnał gotowości nadawania, bądź wybór trybu full–/half–duplex
12	RTS/DIR	Wy	Request To Send. Sygnał żądania nadawania, kontrola kierunku transmisji dla trybu half–duplex.



Rys. 1. Schemat ideowy konwertera

śród których wykorzystamy 6. Opis wszystkich wyprowadzeń modułu przedstawiono w **tab. 1**. Od strony Ethernetu moduł posiada gniazdo RJ45 oraz wbudowany transformator separujący, co znacznie upraszcza projekt płytki drukowanej. Do poprawnej pracy wymagane jest zastosowanie układu generującego właściwy sygnał zerowania tuż po włączeniu zasilania (IC2). Linia MD jest na stałe podciągnięta do zasilania (R3), ale panujący na niej stan może być kontrolowany przy pomocy przycisku S1. Linia ta może być również sterowana przez mikrokontroler. Taki sposób połączenia umożliwia pełną konfiguracje modułu od strony UART–u, przy pomocy zestawu komend. Jeśli taka opcja nas nie interesuje, wówczas linia MD może być na stałe podciągnięta do zasilania, a sam moduł będzie konfigurowalny jedynie od strony Ethernetu za pośrednictwem aplikacji DS Manager dostarczonej przez Tibbo (omówimy ją w dalszej części). Od strony UART–u w użyciu są wyłącznie niezbędne jego linie TX oraz RX doprowadzone do konwertera poziomów IC1, a ich stan jest prezentowany przy pomocy diod LED1 i LED2. Wyjście układu to żeńskie złącze DSUB9. W bloku zasilania zarówno modułu, jak i konwertera poziomów zastosowano liniowy stabilizator napięcia +5 V (IC3) oraz zestaw kondensatorów filtrujących. W przypadku samego modułu EM202 filtrację napięcia zapewnia dodatkowo dławik L1. Dioda D1 zabezpiecza układ przed napięciem o odwrotnej polaryzacji. Całość można zasilać niestabilizowanym źródłem napięcia z przedziału 6 do 8 V oraz wydajności prądowej przynajmniej 300 mA.

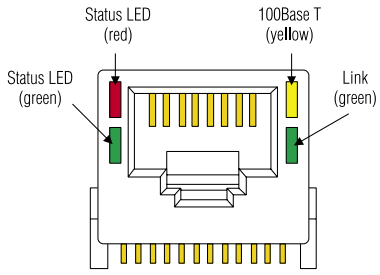
Widoczne na schemacie wyprowadzenia SH1 i SH2 to punkty montażowe obudowy. Jak widać zostały one podłączone do potencjału masy układu. Ciekawostką konstrukcyjną gniazda RJ45 modułu jest to, iż nawet ekranowane wtyki nie są w kontakcie elektrycznym z obudową (co w przypadku podłączenia jej do masy urządzenia, tak jak na schemacie, niszczyłoby separację galwaniczną linku ethernetowego).



Rys. 2. Architektura systemu

Oprogramowanie

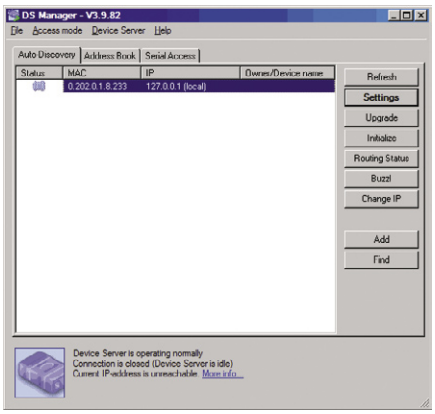
Przy pomocy modułu oraz oprogramowania udostępnionego przez Tibbo zestawimy system, którego schemat blokowy przedstawiono na **rys. 2**. Naszym celem będzie zainstalowanie w systemie operacyjnym sterownika wirtualnego portu szeregowego (wirtualnego w sensie braku obecności układu UART lokalnie w maszynie). Jego fizycznym interfejsem będzie UART modułu EM202 zainstalowanego „gdzieś" w sieci ethernetowej. Z poziomu aplikacji uruchamianej na lokalnym komputerze nie będzie żadnej (prawie) różnicy pomiędzy portami lokalnymi a wirtualnymi (zdalnymi). Aby zrozumieć ograniczenia, jakie wy-



Rys. 3. Diody statusowe modułu umieszczone wokół gniazda RJ45 (za dokumentacją modułu)

stępują w przypadku stosowania wirtualnego portu szeregowego przejdziemy krok po kroku przez wszystkie etapy konfiguracji łańcucha.

W celu rozpoczęcia pracy musimy zaopatrzyć się w oprogramowanie Device Server Toolkit (DST). Jest to zestaw narzędzi obejmujący:



Rys. 4. Okno aplikacji DS Manager

– Device Server Manager – komponent służący do zarządzania, konfiguracji oraz uaktualniania firmware'u modułu,
– Virtual Serial Port Driver (VSPD) – sterownik wirtualnego portu szeregowego,
– Port Monitor – narzędzie pozwalające na monitorowanie stanu portu szeregowego.

DST możemy pobrać ze strony producenta: http://www.tibbo.com/downloads.php (po uprzedniej rejestracji).

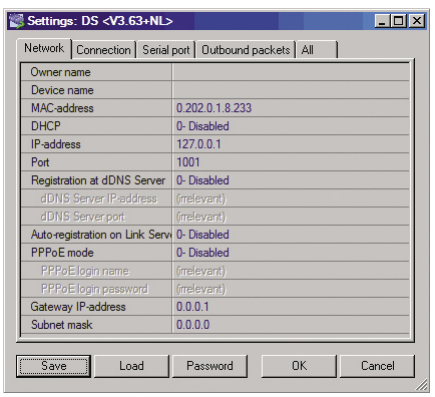
Po rozpakowaniu i instalacji, w menu start pojawi się grupa Tibbo, a w niej Connection Wizzard – kreator połączenia z modułem przeprowadzający użytkownika przez proces konfiguracji, DS Manager, Pliki pomocy, Port Monitor, VSP manager (narzędzie konfigurujące sterownik VSPD) oraz opcja deinstalacji całego zestawu narzędzi. Najpierw podłączmy komputer oraz moduł konwertera do tego samego segmentu sieci (za pośrednictwem switch'a), bądź bezpośrednio (używając skrosowanego kabla ethernetowego). Następnie włączamy zasilanie modułu i obserwujemy diody statusowe (czerwona i zielona) umieszczone po lewej stronie gniazda RJ45 (**rys. 3**). Powinniśmy ujrzeć trzy szybkie błyski obydwu diod. Następnie (po fazie inicjalizacji) powinniśmy zaobserwować dwa cykliczne błyski zielonej diody statusowej, co informuje nas o tym, iż nie ma aktywnego połączenia sieciowego z modułem. Diody po prawej stronie gniazda informują nas z kolei o tym, czy warstwa ethernetowa łącza funkcjonuje poprawnie. Dioda 100BaseT jest włączona na stałe, jeśli uzyskaliśmy połączenie ethernetowe z hubem (jest to urządzenie sieciowe takie jak: switch lub router, jak również karta sieciowa zainstalowana w komputerze) z prędkością 100 Mbit/s. W przeciwnym przypadku (jeśli jest to 10 Mbit/s) dioda jest wyłączona. Z kolei zielona dioda oznaczona, jako Link informuje nas o statusie połączenia (czyli czy druga strona jest gotowa na przesyłanie pakietów). Innymi słowy, czy hub jest włączony czy nie (lub czy warstwa fizyczna – kabel – jest sprawna). Dioda Link przygasa w momentach odbioru przez moduł pakietu Ethernet. Jeśli wszystko działa poprawnie, możemy uruchomić DS Managera (**rys. 4**).

Fabrycznie nowy moduł jest wstępnie skonfigurowany tak, aby włączony w dowolne miejsce dowolnej sieci nie powodował konfliktów. Jest to zestaw dwóch ustawień dotyczących dwóch różnych warstw komunikacji.

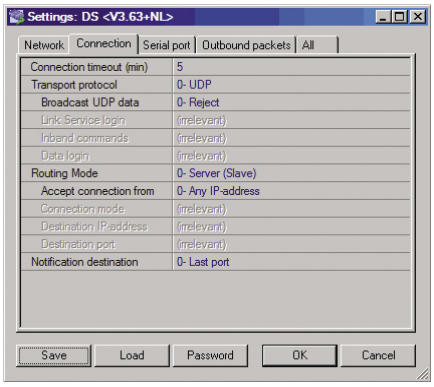
Po pierwsze: każdy moduł EM202 posiada unikatowy adres ethernetowy – MAC (*Media Access Control*), co gwarantuje brak konfliktów pomiędzy urządzeniami w warstwie łącza danych (modelu OSI). Aby być precyzyjnym, należy powiedzieć, że jest to gwarancja jednostronna. Oznacza to, że nigdy nie możemy być pewni, czy druga strona posiada również unikatowy adres ethernetowy, natomiast możemy być pewni tego, że jeśli w war-

Tab. 2. Ikony statusu DS Managera

	Lewa część ikony
	Moduł wysyła zapytanie ARP (Address Resolution Protocol). ARP jest metodą wyszukiwania adresu ethernetowego, kiedy znany jest adres IP hosta.
	Ustanawianie połączenia TCP.
	Połączenie TCP jest ustanowione (lub jest aktualnie zamykane), bądź trwa wymiana danych przy pomocy protokołu UDP. Bufory nadawcze i odbiorcze modułu nie są przepełnione.
	Połączenie TCP jest ustanowione (lub jest aktualnie zamykane), bądź trwa wymiana danych przy pomocy protokołu UDP. Wykryte zostało przepełnienie buforów modułu.
	Połączenie TCP zostało niespodziewanie zakończone przez zdalnego hosta.
	Środkowa część ikony
	Brak informacji na temat statusu. Ikona ta oznacza, że w module znajduje się stara wersja firmware'u (2.xx lub starsza).
	Moduł jest widoczny w sieci i pracuje poprawnie.
	Moduł pracuje w trybie awaryjnym i wymaga inicjalizacji, która przywróci wszystkie domyślne ustawienia modułu poza adresem IP.
	Moduł nie zdołał pobrać swojej konfiguracji z serwera DHCP (oraz może znajdować się w trybie awaryjnym). Status taki może wystąpić tylko w przypadku uaktywnienia klienta DHCP modułu.
	Moduł znajduje się w trybie uaktualnienia firmware'u. Jeśli taki status pojawia się tuż po starcie, może oznaczać to, że moduł nie posiada poprawnego obrazu firmware'u lub firmwar'e jest uszkodzony.
	Prawa część ikony
	Ikona pojawia się w momencie zapisywania ustawień konfiguracyjnych modułu, bądź w trakcie uaktualniania firmawe'u.



Rys. 5. Zakładka ustawień sieciowych modułu



Rys. 6. Zakładka ustawień połączeń modułu

stwie fizycznej wystąpi konflikt, to nie będzie on wywołany przez moduł EM202 z oryginalnymi ustawieniami adresu MAC.

Po drugie: każdy moduł EM202 posiada wstępnie przypisany adres IP: 127.0.0.1 (tzw. adres *loopback*). Ustawienie to gwarantuje brak konfliktów w warstwie sieciowej. Z drugiej strony, tak skonfigurowany moduł nie ma większego praktycznego zastosowania.

Na rys. 4 przedstawiono okno programu tuż po operacji autowyszukiwania modułów (zakładka Auto–Discovery). Operacja ta może zakończyć się sukcesem tylko wtedy, gdy zastosujemy taką topologię połączeń, jak opisana wyżej (jeśli moduł byłby podłączony do segmentu sieci oddzielonego od naszego komputera, np. routerem, wówczas moduł nie zostałby wykryty). W każdym momencie działania programu możemy kliknąć na przycisk *Refresh*, co spowoduje odświeżenie listy widocznych modłów. Aby upewnić się, iż komunikacja z konkretnym modułem przebiega prawidłowo (bądź, jeśli chcemy zidentyfikować moduł) możemy skorzystać z opcji Buzz!, która spowoduje, iż statusowe diody będą zapalać się naprzemiennie przez około 1 sekundę. W oknie DS Managera widoczne są adresy MAC i IP modułu, jak również stratus połączenia (w **tab. 2** przedstawiono opis statusów połączenia i odpowiadających im ikon).

Przejdźmy do etapu konfiguracji modułu. W oknie DS Managera zaznaczamy nasz moduł, po czym klikamy przycisk *Settings* W nowo otwartym oknie domyślnie znajdujemy się w pierwszej zakładce ustawień związanych z obsługą interfejsu sieciowego (**rys. 5**). W naszym przypadku nie będziemy musieli modyfikować wszystkich ustawień i skupimy się jedynie na tych niezbędnych do poprawnej pracy modułu w charakterze wirtualnego portu szeregowego. Są one opisane niżej.

DHCP. Opcja uaktywniająca klienta DHCP modułu. Domyślnie przyjmuje wartość zero. Takie też ustawienie jest prawidłowe w przypadku pracy ze sterownikiem wirtualnego portu szeregowego.

IP–address. Adres IP modułu. Należy nadać modułowi adres IP pochodzący z tej samej podsieci, co adres komputera, na którym zainstalowany jest sterownik portu. Np. dla adresu komputera 192.168.1.2/ 255.255.255.0, możemy nadać modułowi adres IP 192.168.1.10 (jeśli nie jest on zajęty).

Port. Port (lokalny modułu) protokołu TCP bądź UDP, poprzez który będzie odbywać się komunikacja modułu ze sterownikiem. Wartość ta może pozostać niezmieniona

Gateway IP address. Adres IP bramy. W przypadku pracy w tym samym segmencie sieci parametr ten nie jest istotny.

Subnet mask. Maska podsieci. Ustawienie to zależy od lokalnej konfiguracji segmentu sieci. W naszym przypadku będzie to 255.255.255.0

Przejdźmy do zakładki *Connection*, w której ustalimy, w jaki sposób, będziemy transportować dane z i do modułu (**rys. 6**). Mamy tu opcje opisane niżej.

Connection timeout (min). Czas, po którym nieaktywne połączenie sieciowe zostanie przerwane. Ustawienie ma praktyczne znaczenie w przypadku połączenia TCP.

Transport protocol. Pole wyboru protokołu warstwy transportowej, jaki będzie użyty do komunikacji modułu z hostem. Do wyboru mamy oczywiście protokoły TCP oraz UDP. Zależnie od naszego wyboru, pojawiają się dodatkowe opcje:

TCP

– *Link Service login* – ustawienie dotyczące pakietu oprogramowania *LinkServer*. Omówienie go wykracza poza ramy artykułu.
– *Inband commands* – uaktywnia, bądź blokuje możliwość konfiguracji modułu przy pomocy specjalnego zestawu komend, które mogą być przesyłane do modułu w trakcie sesji danych TCP.
– *Data login* – ustawienie definiuje czy moduł ma wejść w tryb konfiguracji, akceptując wyłącznie komendy.

UDP

– *Broadcast UDP data* – opcja określająca czy moduł ma akceptować datagramy UDP, które dotarły do modułu w trybie broadcast'u, a więc bez adresu odbiorcy. Opcja ta jest przydatna wtedy, gdy chcemy wysłać identyczne dane do wszystkich modułów naraz.

Wybierzmy TCP jako protokół transportowy. Pozostałe ustawienia mogą pozostać domyślne.

Routing Mode – Parametr ten definiuje sposób przesyłania danych pomiędzy końcowymi węzłami całego systemu, a więc UART–em modułu oraz sterownikiem wirtualnego portu. Mamy do dyspozycji trzy opcje implikujące kolejne ustawienia:

Server

Stroną inicjalizującą połączenie sieciowe (w naszym przypadku będzie to TCP) jest komputer PC. Jeśli sterownik odbierze znaki, które mają być przesłane do konkretnego portu COM, wówczas zostanie nawiązane połączenie TCP z modułem. Podczas jego trwania mogą być przesyłane dane zarówno z aplikacji do modułu (jego UART–u) oraz w drugą stronę. Połączenie zostanie zamknięte, jeśli tak zadecyduje sterownik VSPD (np. jeśli zakończy się działanie aplikacji), bądź w przypadku, gdy upłynie czas bezczynności określony przez parametr *connection timeout*. Ważna

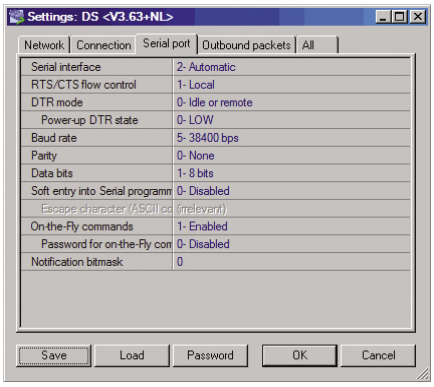
WYKAZ ELEMENTÓW

Rezystory
R1, R2: 310 Ω
R3: 10 kΩ

Kondensatory
C1, C2, C3, C4, C5, C6: 100 nF
C7, C8: 1 µF/25 V

Półprzewodniki
LED1, LED2: Dioda LED
D1: MBRS130LT3 Schottky 1 A, 30 V
IC1: MAX202
IC2: MAX810
IC3: XC6203
Stabilizator liniowy +5 V
EM1: Moduł Tibbo EM202

Inne
L1: Filtr 1000Z@100 MHz
X1: D–SUB9
J1: DC JACK 2 mm
S1: Mikrowyłącznik



Rys. 7. Zakładka ustawień układu UART modułu

jest świadomość tego, iż w tym trybie urządzenie końcowe podłączone do UART–u modułu konwertera, nie będzie w stanie wysyłać znaków (danych) do komputera PC, dopóki ten nie nawiąże połączenia TCP z modułem.

Innymi słowy, urządzenie to będzie w systemie urządzeniem podrzędnym (*slave*). Po wybraniu tej opcji, przy pomocy parametru: *Accept connection from* możemy zdecydować czy moduł ma akceptować połączenia przychodzące z jakiegokolwiek adresu IP (ustawienie *Any IP–address*), czy chcemy wyspecyfikować tylko jeden adres hosta, który będzie w stanie nawiązać połączenie TCP z modułem (*IP matching destination IP–address*). Jeśli wybierzemy taką opcję, to w polu *Destination IP–address* musimy wpisać adres IP komputera, na którym zostanie zainstalowany sterownik VSPD.

Server OR Client

Najbardziej uniwersalny tryb. VSPD może nawiązać połączenie z modułem, jak i moduł może nawiązać połączenie z hostem podczas odbioru danych przez UART. Oczywiście w tym przypadku musimy podać zestaw parametrów komputera docelowego: *Destination IP–address, Destination port*. Tak jak poprzednio mamy możliwość włączenia filtrowania połączeń przychodzących. Najbardziej krytycznym ustawieniem jest definicja zdarzenia, po którym moduł ma nawiązać próbę ustanowienia połączenia TCP. Parametr ten nosi nazwę *Connection mode* i może przyjmować cztery wartości: *Immediately (on powerup)* – próba nawiązania połączenia z hostem nastąpi tuż to włączeniu modułu, *On data OR command* – moduł nawiąże połączenie TCP tuż po odebraniu przez UART pierwszego bajtu lub komendy ustanowienia połączenia (również przez układ UART), *On command only* – połączenie i transmisja danych odebranych przez UART nastąpi jedynie po odbiorze przez moduł komendy ustanowienia połączenia, *On command or DSR=HIGH* – połączenie zostanie ustanowione po wydaniu komend, bądź po detekcji stanu wysokiego na linii DSR modułu.

Najbardziej uniwersalną opcją (przynajmniej w przypadku naszego konwertera) jest *On data OR command*.

Client only

W tym trybie jedynie moduł może inicjować połączenia sieciowe z hostem. Warunki ustanowienia połączenia wyglądają tak samo jak w trybie *Server OR Client*.

Na **rys. 7** przedstawiono widok zakładki ustawień układu UART modułu. Oprócz standardowych wartości takich jak prędkość bodowa (*Baudrate*), bit parzystości (*Parity*), czy liczba bitów jednej ramki danych (*Data bits*), mamy do dyspozycji kilka ustawień (związanych np. z kontrolą przepływu), które bezpośrednio wpływają na sposób działania modułu. Opisano je niżej.

Serial interface. Parametr ten decyduje o logicznym (nie sprzętowym) sposobie działania układu UART. Może przyjmować takie wartości jak Full–duplex, Half–duplex, Automatic.

Pierwszy wariant jest przeznaczony dla współpracy modułu z transceiverami RS232 (tak jak w przypadku naszego konwertera) i RS422. W tym trybie do dyspozycji mamy dwie standardowe linie kontroli przepływu: RTS oraz CTS (oczywiście nie musimy z nich korzystać i w przypadku konwertera tak właśnie się dzieje). Tryb pół–dupleksowy przeznaczony dla komunikacji w standardzie RS485. Linia RTS zapewnia kontrolę kierunku transmisji, podczas gdy CTS nie jest używana. My zastosujemy pierwszą opcję lub opcję Automatic, kiedy to selekcja trybu pracy jest dokonywana automatycznie na podstawie zewnętrznych połączeń linii modułu (dla selekcji trybu automatycznego fizyczne linie CTS/SEL oraz SR powinny pozostać niepołączone).

RTS/CTS flow control. Ustawienie sprzętowej kontroli przepływu. Do wyboru mamy jej brak – Disabled or remote, bądź Local – kiedy to sprzętowo (np. z poziomu mikrokontrolera) musimy sterować linią CTS oraz sprawdzać stan linii RTS. Dla konwertera zablokujmy sprzętową kontrolę przepływu.

DTR. Linia ta może być skonfigurowana w dwojaki sposób:

- **Indicate connection status** – poziom logiczny linii będzie wskazywał aktywne połączenie sieciowe (lub jego brak),
- **Idle or remote** – stan linii będzie kontrolowany zdalnie z poziomu hosta.

W naszym przypadku ustawienie tego parametru nie ma żadnego znaczenia, gdyż linia ta nie jest podłączona.

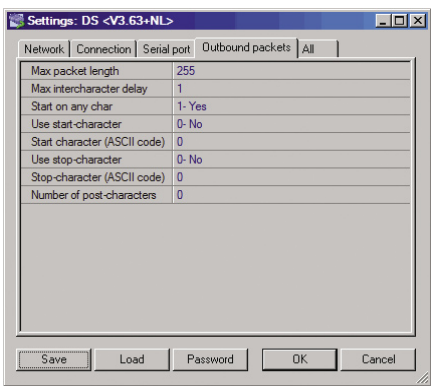
Soft entry into Serial programming mode. Opcja służy do wprowadzania modułu w stan programowania (zmiany parametrów pracy) przy pomocy specjalnego zestawu komend. Możemy ją zablokować (Disabled), bądź uaktywnić w jednym z dwóch sposobów wprowadzania modułu w stan programowania (są to dwie róże sekwencje znaków). Uaktywnienie tej opcji prowadzi za sobą obowiązek kontrolowania danych wysyłanych do modułu poprzez UART tak, aby uniknąć przypadkowej zmiany trybu pracy modułu. Można to postrzegać jako pewne ograniczenie, gdyż urządzenie nie jest od tego momentu przeźroczyste dla transportowanych danych. W przypadku konwertera zablokujemy tę opcję.

On–the–fly commands. Kolejna opcja zapewniająca (jeśli jest włączona – *Enabled*) możliwość zmiany nastaw modułu poprzez zestaw komend. Tym razem komendy mogą być zaszyte w transportowanej paczce danych i opatrzone specjalnymi znacznikami (szczegóły znajdują się w dokumentacji modułu). W przypadku uaktywnienia tej opcji dane wysyłane do modułu nie mogą przypadkowo przybrać postaci jakiejkolwiek komendy. Należy zachować ostrożność w jej stosowaniu.

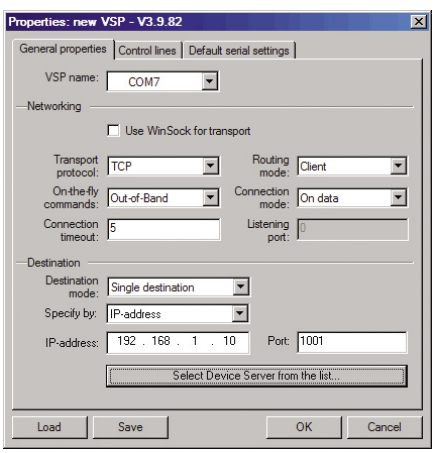
Pozostał nam do omówienia ostatni z zestawów konfigurowalnych parametrów modułu – *Outbound packets* (**rys. 8**). Ustawienia te decydują o sposobie enkapsulacji danych pochodzących z UART–u w pakiety ethernetowe. Prawidłowa konfiguracja tych parametrów nie jest trywialna, ponieważ balansujemy pomiędzy obciążeniem sieci (dużo małych pakietów), a prędd-

kością transmisji (kiedy zdecydujemy się na transmitowanie dłuższych ramek, wówczas kolekcjonowanie ich poszczególnych bajtów trwa dość długo, spowalniając w rezultacie całą transmisję w łańcuchu).

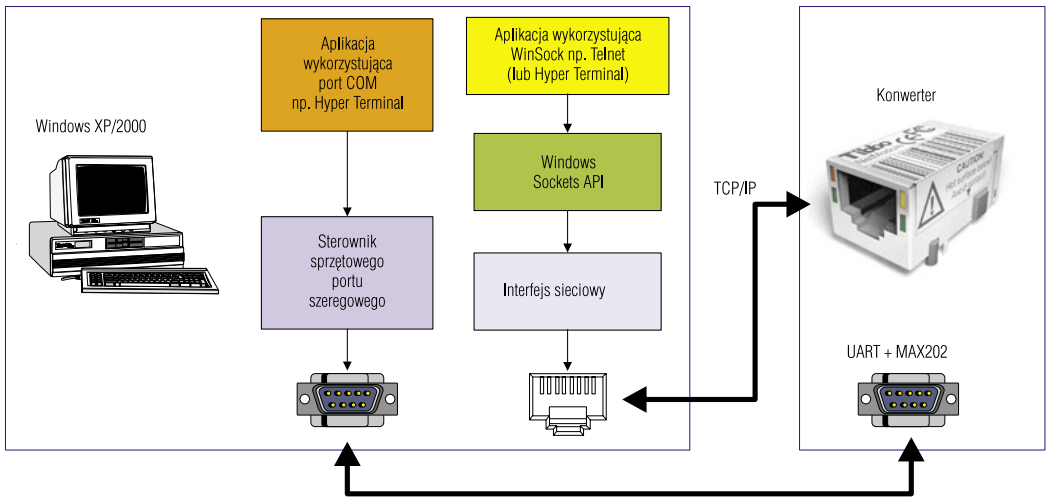
Max packet length. Max intercharacter delay. Są to dwa oddzielne parametry, ale na tyle ze sobą skorelowane, że należy omówić je razem. Pierwszy z nich definiuje długość kolekcjonowanej ramki danych przychodzących z UART–u, która zostanie nadana przy pomocy wybranego protokołu transportowego. Aby zbieranie ramki nie trwało jednak zbyt długo, wprowadzono drugi parametr czuwający nad tym, aby ramka została nadana w przypadku zbyt długiego oczekiwania na kolejne znaki. Innymi słowy: jest to maksymalny odstęp pomiędzy kolejnymi bajtami, który nie powoduje wysłania ramki protokołem transportowym.



Rys. 8. Zakładka ustawień enkapsulacji pakietów



Rys. 9. Konfiguracja nowego portu COM



Rys. 10. Komunikacja z modułem przy użyciu standardowego mechanizmu WinSock

Start on any char. Parametr definiuje czy ramka danych, która zostanie nadana poprzez sieć ma rozpoczynać się dowolnym znakiem odebranym przez UART, czy też jako pierwszy musi zostać odebrany konkretny bajt.

Use start–character. Jeśli poprzedni parametr posiada wartość 0 (No), wówczas musimy zdefiniować, tzw. znak startu i parametr musi przybrać wartość 1.

Start–character (ASCII code). Jeśli poprzedni parametr jest równy 1, wówczas musimy wyspecyfikować konkretny znak startu, którego kod ASCII będzie wartością obecnego parametru.

Use stop–character. Parametr analogiczny do Use start–character, definiuje czy kolekcjonowanie ramki danych ma zakończyć się w momencie nadejścia konkretnego znaku.

Stop–character (ASCII code). Kod ASCII znaku stopu (jeśli Use stop–character =1)

Number of post–characters. Parametr definiuje liczbę znaków odebranych przez UART po wystąpieniu znaku stopu, które ciągle będą wliczone do bieżącej ramki.

Jeśli moduł ma być używany jako wirtualny port szeregowy (a więc ma być przeźroczysty dla wszelkich danych), wówczas wszystkie parametry tej sekcji powinny przybrać swoje wartości domyślne (tak jak przedstawiono to na **rys. 9**).

Mając zgrubny obraz tego, co będzie działo się w sieci pomiędzy konwerterem a komputerem, możemy świadomie przystąpić do instalacji sterownika VSPD.

Wirtualny port szeregowy

Uruchamiamy VSP Managera. W głównym oknie przedstawia on listę (jak na razie pustą) obecnie zainstalowanych portów wirtualnych. W celu zdefiniowania nowego portu klikamy na przycisk *Add*, co spowoduje wyświetlenie okna konfiguracji nowego portu (rys. 9).

Po obszernym omówieniu opcji ustawień modułu, jakie oferował DS Manager, konfiguracja sterownika wirtualnego portu nie powinna przynieść problemów. Do wyboru pozostała nam tak naprawdę jego nazwa oraz podanie adresu IP modułu oraz numeru portu, na którym ma odbywać się komunikacja. Po kliknięciu *OK* program dokona właściwej instalacji sterownika w systemie, a manager urządzeń powinien pokazać nowy port COM. Dodatkowo, poprawną instalację VSPD możemy zweryfikować zaglądając do sekcji Porty COM i LPT managera urządzeń systemu.

Alternatywne metody dostępu do modułu

Moduł EM202 sam w sobie nie ma percepcji tego czy dane wysyłane do niego pochodzą ze sterownika VSPD, dlatego możemy używać go z jakąkolwiek aplikacją potrafiącą nawiązywać połączenia TCP (bądź UDP). Pierwsza, jaka naturalnie przychodzi na myśl to telnet uruchamiana z konsoli systemowej. Wpiszmy, zatem w linii komend systemu Windows:

```
telnet 192.168.1.10 1001
```

gdzie 192.168.1.10 to IP modułu, a 1001 to port na którym moduł oczekuje na połączenia TCP. Po nawiązaniu połączenia powin-

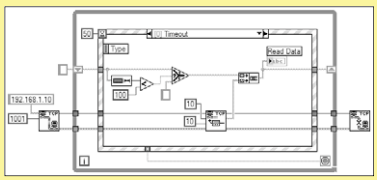
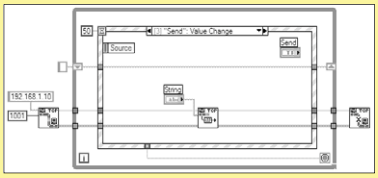
niśmy obserwować stałe świecenie zielonej diody statusowej modułu, a wpisując przy pomocy klawiatury jakikolwiek ciąg znaków zauważymy błyski diody LED1 podłączonej do linii TX UART–u. Idąc dalej, podłączając prostym (w sensie przyporządkowania pinów) kablem port RS232 konwertera z portem RS32 w innym komputerze (lub nawet w tym samym) i nasłuchując przychodzące do niego znaki (np. przy pomocy HyperTerminala) powinniśmy widzieć to, co wpisujemy w otwartej sesji telnet. Sytuację tę przedstawiono na **rys. 10**.

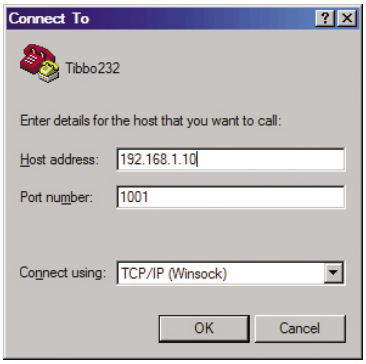
Zamiast konsolowej aplikacji, jaką jest Telnet do nawiązywania połączenia TCP możemy użyć HyperTerminala. Na etapie konfiguracji połączenia w polu „Połącz używając", z rozwijanej listy należy wybrać opcję TCP/IP (Winsock), a następnie skonfigurować odpowiednio sesję. Podejście takie pozwala nam korzystać ze wszelkich dogodności, jakie daje HyperTerminal, gdyż z punku widzenia użytkownika program zachowuje się tak, jakby korzystał z lokalnego portu COM (**rys. 11**).

Jednak najbardziej uniwersalnym i wygodnym sposobem na komunikację z modułem konwertera jest stworzenie własnej aplikacji. Posługując się środowiskiem LV napiszemy prosty program, który będzie nadawać do modułu dowolny podany przez użytkownika ciąg znaków oraz permanentnie odbierać jakiekolwiek dane napływające z modułu. Wykorzystamy tylko jeden z możliwych scenariuszy komunikacyjnych. Komputer PC będzie zawsze stroną inicjalizującą połączenie TCP. Łańcuchy znaków wpisywane przez użytkownika będą na żądanie nadawane do modułu, a ten (bez sprzętowej kontroli przepływu) będzie przekazywał je do swojego układu UART. W drugą stronę: jeśli podczas otwartego połączenia TCP, UART modułu odbierze jakiekolwiek znaki, to natychmiast zostaną one użyte do stworzenia segmentu TCP i zostaną nadane do hosta. Realizujący powyższe założenia kod LV przedstawiono w **tab. 3**.

Prezentowany program VI jest bardzo prosty i pokazuje jedynie pryncypia stosowania modułu konwertera z własnymi aplikacjami. Dla czytelników chcących pogłębić umiejętności stosowania LV proponuję takie zmodyfikowanie kodu programu (i ustawień modułu), aby do komunikacji z nim był wykorzystywany protokół UDP.

Tab. 3. Kod LV obsługujący transmisję przez moduł konwertera Ethernet<–> RS23

	Komponent TCP Open Connection odpowiedzialny jest za ustanowienie połączenia. Jak widać ma na stałe przypisane wartości adresu IP modułu oraz portu (jedynie na potrzeby przykładu).
	Po poprawnej inicjalizacji program przechodzi do swojej pętli głównej, w której oczekuje na zdarzenia generowane przez użytkownika na panelu frontowym aplikacji (**rys. 12**). Jeśli zdarzenia te nie mają miejsca (oczekujemy na nie przez 50 ms), wówczas wykonuje się widoczny na diagramie przypadek: odczyt 10 bajtów z bufora TCP. Jeśli w buforze nie ma wystarczające ilości danych, komponent TCP Read oczekuje na ich nadejście przez 10 ms, po czym zwraca wszystkie zebrane do tej pory dane, które są koncentrowane w jeden ciąg znaków (razem z danymi z poprzedniej iteracji) i prezentowane w indykatorze Read Data.
	Jeśli użytkownik wygeneruje zdarzenie polegające na kliknięciu przycisku Send, wówczas jego obsługa sprowadzi się do wysłania do modułu zawartości kontrolki Data to Send.
	W przypadku detekcji zdarzenia polegającego na kliknięciu przycisku Exit pętla programu zostanie zatrzymana i zostanie wywołany komponent TCP Close Connection, który zamknie połączenie TCP z modułem.

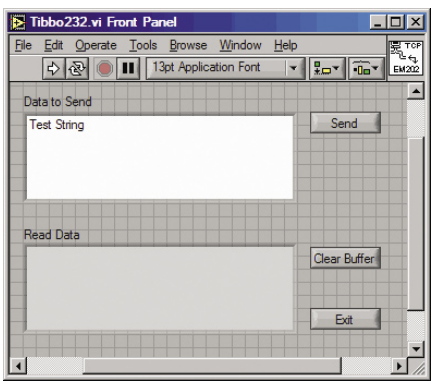


Rys. 11. Sesja HyperTerminala nawiązująca połączenie TCP/IP przy pomocy WinSock

Podsumowanie

Moduł konwertera zaprezentowany w artykule może stanowić platformę do weryfikacji przydatności modułu EM202 w nowo projektowanym urządzeniu. O ile idea jego konstrukcji jest dość prosta, to pewne niuanse konfiguracyjne dają poczucie tego, że nie jest to urządzenie banalne. Samodzielne poznanie metod komunikacji pozwoli natomiast prawidłowo ocenić przydatność konstrukcji w konkretnym zastosowaniu. Stosując standardowy firmware modułu możemy bardzo szybko dokonać uaktualnienia istniejących urządzeń wyposażonych w RS232, czy chociażby UART–u do urządzeń stricte sieciowych. Z drugiej strony, moduł EM202 nie jest jedynym sposobem na podłączenie urządzenia mikroprocesorowego do sieci ethernetowej, jednak wsparcie przez stworzone wokół niego oprogramowanie, jak również świetna dokumentacja czynią je wyjątkowo wydajnym.

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



Rys. 12. Panel frontowy aplikacji