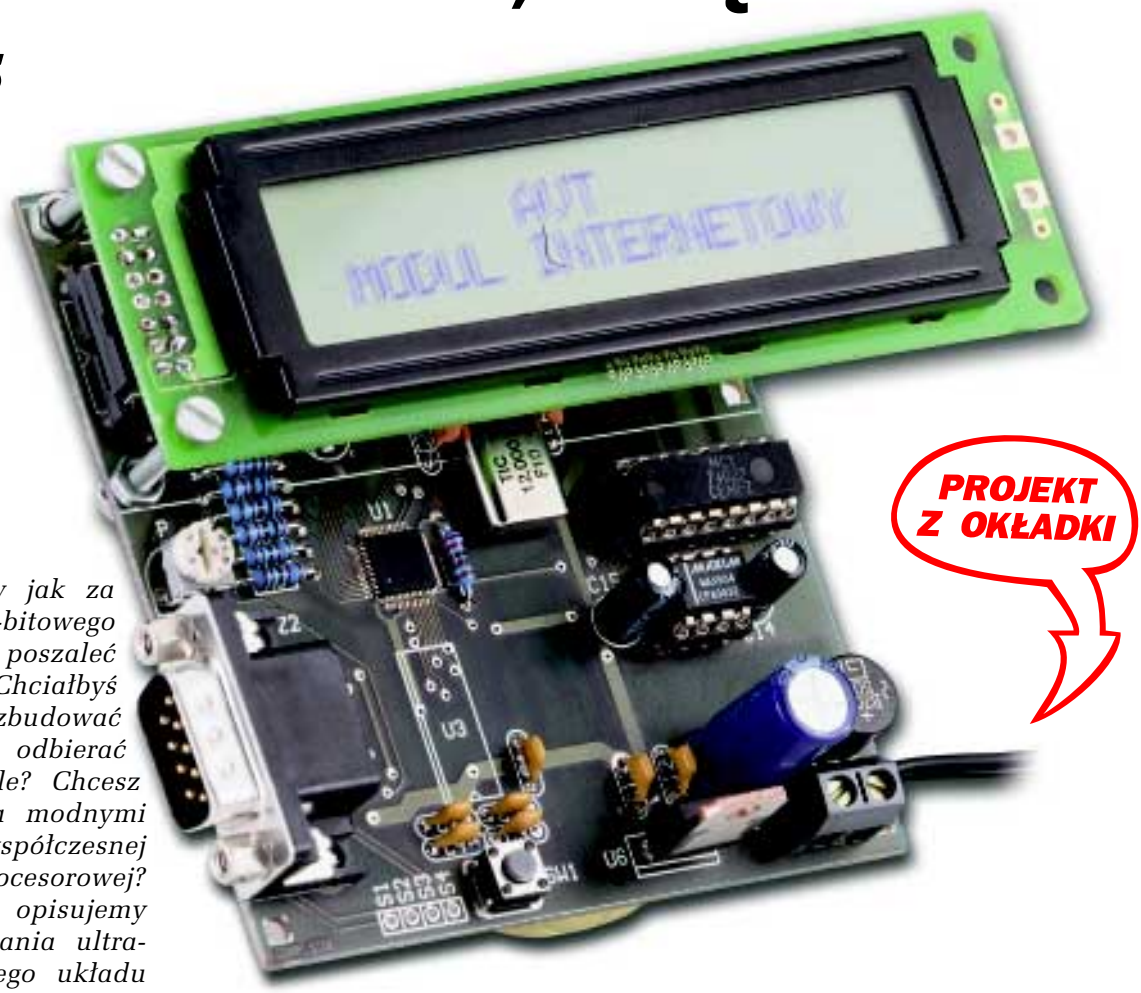
Internetowy interfejs dla mikrokontrolera, część 1

AVT-5055



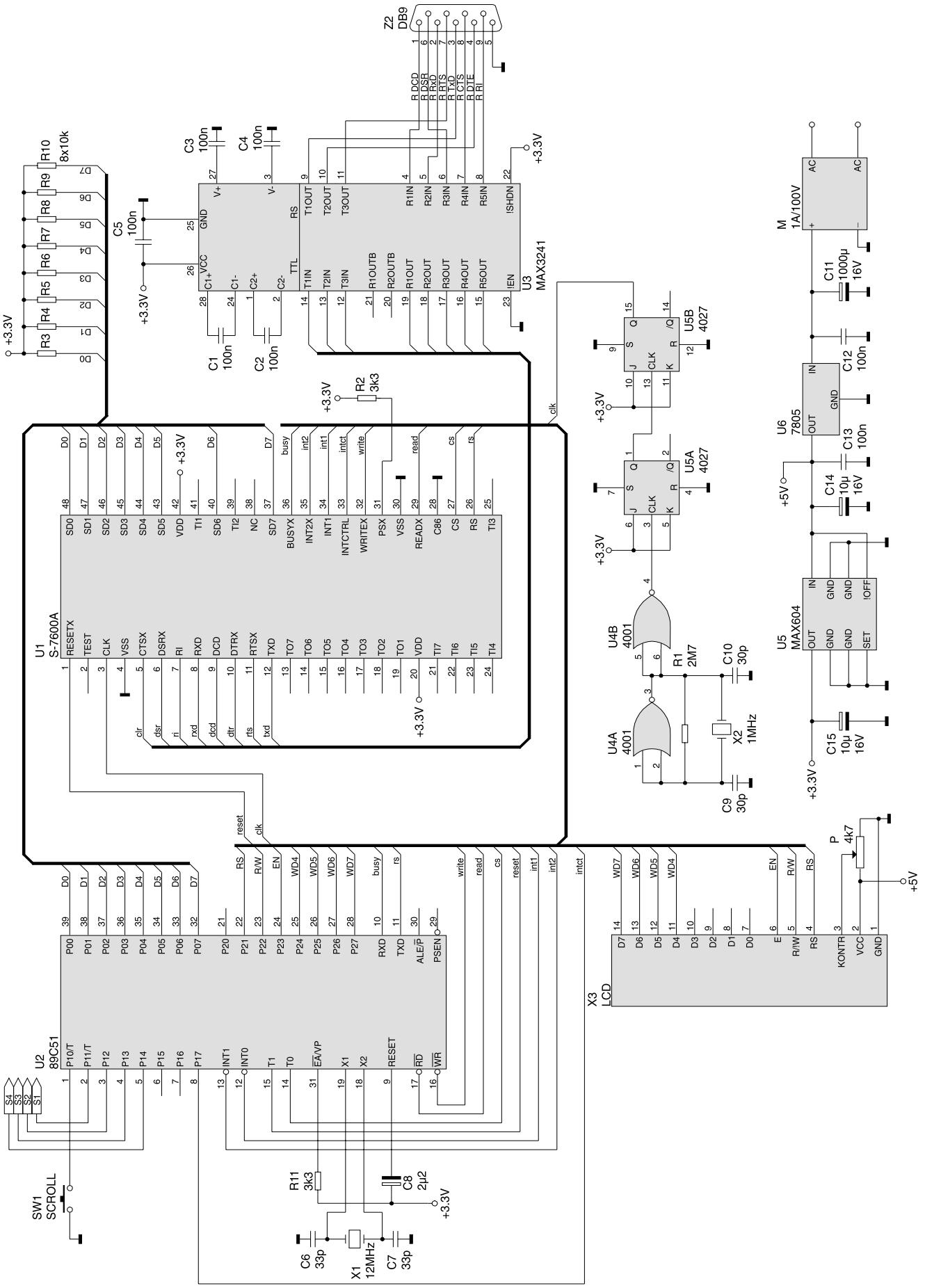
PROJEKT Z OKŁADKI

Jesteś ciekaw jak za pomocą 8-bitowego mikrokontrolera poszaleć w Internecie? Chciałbyś samodzielnie zbudować serwer internetowy, odbierać i nadawać e-maile? Chcesz nadążyć za modnymi zakamarkami współczesnej techniki mikroprocesorowej? W artykule opisujemy sposób wykorzystania ultranowoczesnego układu scalonego, który spełnia rolę sprzętowego stosu TCP/IP.

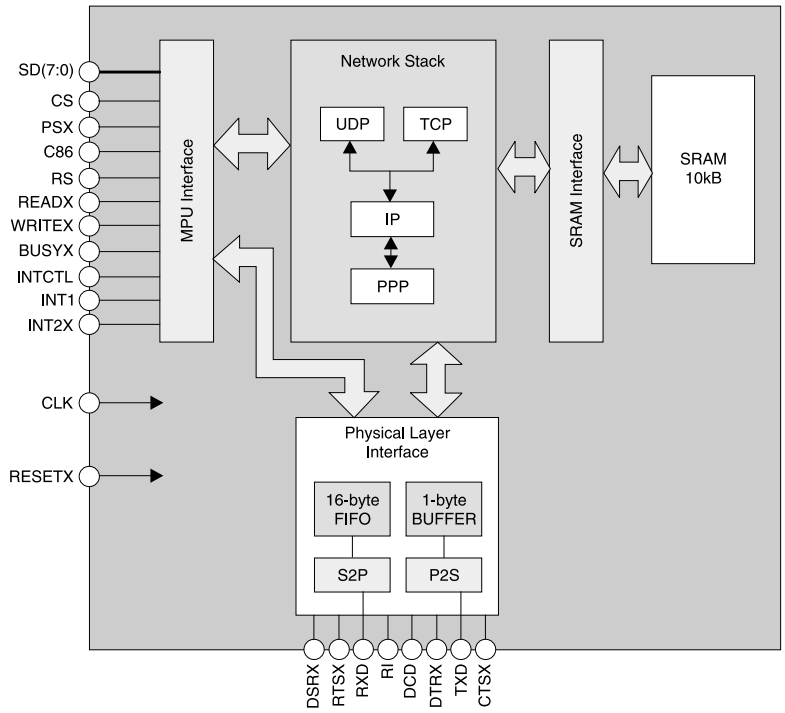
O możliwości wykorzystania Internetu do przesyłania danych przez mikroprocesorowe sterowniki słychać coraz częściej i coraz głośniej. Firmy produkujące mikrokontrolery oferują zestawy ewaluacyjne pozwalające na połączenie się z Siecią - większość z nich opisywaliśmy już w EP. Na przykład takimi spektakularnymi aplikacjami mogą być opisywane już opracowania Atmela, Ziloga, Microchipa czy firmy Ubicom (dawniej Scenix). W tym ostatnim wykorzystano nowo opracowane mikrokontrolery z odpowiednio wydajnym rdzeniem oraz „dużą" pamięcią danych i programu. Przy takiej koncepcji, obsługa wszystkich protokołów sieciowych niezbędnych do połączenia z Internetem i przesyłania informacji realizowana jest programowo.

Takie rozwiązanie ma oczywiście wiele zalet. Po poznaniu modułów programowych systemu, można w miarę elastycznie dostosowywać go do własnych wymagań. Jednak z drugiej strony, implementacja sieciowa zajmuje dużo mocy obliczeniowej i pamięci mikrokontrolera. Nie bez znaczenia jest też fakt, że trzeba stosować nowe mikrokontrolery, a to wiąże się zawsze z wprowadzeniem nowych narzędzi (kompilatorów, programatorów, emulatorów itp.), a więc ze zwiększeniem kosztów.

Cóż zatem zrobić, aby nie narazić się na dodatkowe koszty i nie popaść w uzależnienie od konkretnego mikrokontrolera? Od tych dylematów uwalnia nas oferta firmy Seiko - układ S7600A. Jest to specjalizowany układ



Rys. 1. Schemat elektryczny interfejsu.



Rys. 2. Schemat blokowy układu S7600A.

umożliwiający szybkie i bezproblemowe połączenie z siecią, praktycznie dowolnego, obecnie produkowanego mikrokontrolera. Przykładem niech będzie opisywany tutaj system, w którym doskonale wszystkim znany i niezbyt wydajny mikrokontroler AT89C51 wraz z układem S7600A pracuje jako klient poczty elektronicznej.

Opis układu

Schemat interfejsu pokazano na **rys. 1**. Najważniejszym jego elementem jest oczywiście S7600A - układ wielkiej skali integracji (VLSI) zawierający w swojej strukturze kompletny, sprzętowy stos TCP/IP wraz z zaimplementowanym protokołem PPP, interfejs łącza szeregowego UART z 16-bitowym odbiorczym buforem FIFO, 10kB pamięci RAM oraz rozbudowany szeregowo - równoległy interfejs MPU do połączenia z mikrokontrolerem (**rys. 2**). Przez ten interfejs mikrokontroler zapisuje lub odczytuje informacje do/z wewnętrznych rejestrów S7600A.

Układ zasilany jest napięciem o wartości z zakresu 2,4V...3,6V i pobiera minimalny prąd w czasie pracy: 0,9mA w trakcie transmisji i tylko 150µA w stanie oczekiwania na transmisję. Tak niski pobór mocy wskazuje na to, że konstruktorzy przewidywali jego pracę przy zasilaniu bateryjnym. Układ jest taktowany zewnętrznym sygnałem zegarowym. Producent zaleca częstotliwość 256KHz, ale maksymalna jej wartość może wynosić nawet 5MHz.

Układ S-7600A zawiera dwa interfejsy MPU: równoległy i szeregowy. W trybie interfejsu równoległego można połączyć magistrale danych rodziny x80 firmy Intel lub 68K firmy Motorola. Poprzez te interfejsy następuje wymiana informacji pomiędzy mikrokontrolerem a układem. Jak widać zadbano, aby maksymalnie ułatwić pracę projektantom. Wybór interfejsu równoległego następuje poprzez wymuszenie wysokiego poziomu na wejściu PSX. Poziom wysoki na wejściu C86 określa tryb pracy interfejsu dla procesorów Motoroli, a poziom niski dla procesorów Intela. W **tab. 1** zawarto zestawienie stanów logicznych na wejściach sterujących zapisem i odczytem dla obu rodzajów magistrali.

Tab. 1. Sposób obsługi rejestrów układu S7600A dla dwóch możliwych konfiguracji interfejsu.

RS	Motorola	Intel		Funkcja
	R/WX	READX	WRITEX	
1	1	0	1	Czytanie rejestru
1	0	1	0	Zapis rejestru
0	1	0	1	Czytanie rejestru indeksowego
0	0	1	0	Zapis rejestru indeksowego

Ponieważ w projekcie wykorzystywana jest magistrala Intela, to ją postaram się opisać dokładniej. Zainteresowani sterowaniem S7600A poprzez magistralę Motoroli mogą znaleźć odpowiednie dane w dokumentacji firmowej.

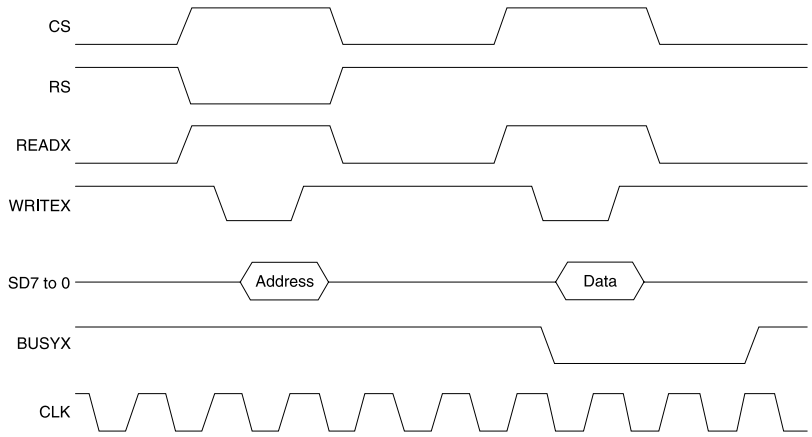
Tryb magistrali Intela jest wprowadzany, kiedy na wejściu C86 jest poziom niski „L“, a na PSX poziom wysoki „H“. Dane oraz magistrala adresowa są multipleksowane. Każdy cykl rozpoczyna się od ustawienia na magistrali adresu. Adres ten jest zatrzaskiwany w wewnętrznym rejestrze podczas narastającego zbocza WRITEX. Poziom niski na RS wskazuje, że strobowanie WRITEX dotyczy fazy adresu na magistrali. W następnej fazie dane mogą być zapisywane lub odczytywane poprzez wygenerowanie odpowiednich zboczy sygnałów WRITEX lub

List. 1. Procedura zapisu danych do układu S7600A przez magistralę równoległą.

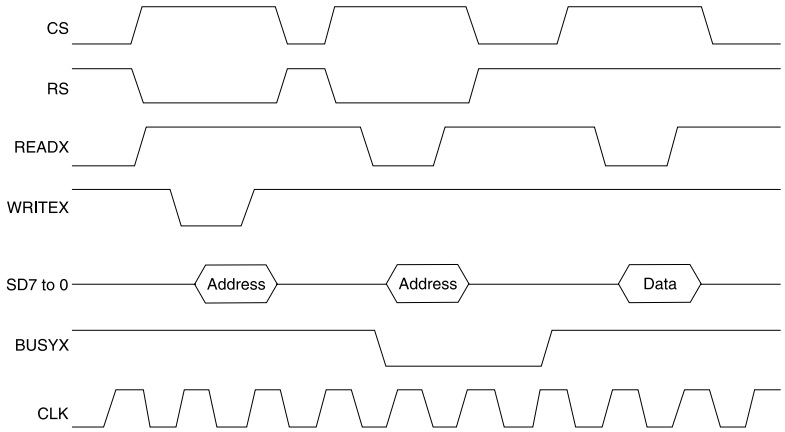
```
void zapis_ichip    unsigned char adres,
                    unsigned char dana)
{
   cs=1;
   rs=0;          //adres na magistrali
   readx=1;
   writex=0;
   P0=adres;      //adres na magistrale
   writex=1;
   cs=0;
   rs=1;          //dane na magistrali
   readx=0;
   cs=1;
   readx=1;
   writex=0;
   P0=dana;       //dana na magistrale
   writex=1;
   cs=0;
   readx=0;
   while(busyx==0);
                  //czekaj na nieaktywne busyx
}
```

List. 2. Procedura odczytu danych z układu S7600A przez magistralę równoległą.

```
unsigned char odczyt_ichip
            (unsigned char adres)
{
unsigned char dana;
unsigned char dana_p;
   cs=1;
   rs=0;
   readx=1;
   writex=0;
   P0=adres;
   writex=1;      //adres wpisany do
   cs=0;
   rs=1;
   cs=1;
   rs=0;
   P0=0xff;       //ustaw jako wejsciowy
   readx=0;
   dana_p=P0;     //odczytanie adresu
   readx=1;
   rs=1;
   cs=0;
   while(busyx==0);
                  //czekaj na nieaktywne busyx
   cs=1;
   readx=0;
   dana=P0;       //odczytanie danych
   readx=1;
   while(busyx==0);
                  //czekaj na nieaktywne busyx
   cs=0;
   return(dana);
}
```



Rys. 3. Przebiegi czasowe sygnałów podczas zapisu do interfejsu równoległego.



Rys. 4. Przebiegi czasowe sygnałów podczas odczytu do interfejsu równoległego.

READX. Układy logiczne interfejsu generują w tej fazie sygnał BUSYX po opadającym zboczu sygnału WRITEX lub READX. Sygnał ten staje się nieaktywny, gdy S7600A zakończy operację zapisu lub odczytu. Mikrokontroler powinien próbkować sygnał BUSYX. Może on zainicjować kolejny cykl zapisu/odczytu dopiero wtedy, gdy BUSYX staje się nieaktywny.

Przebiegi czasowe dla zapisu i odczytu przez magistrale równoległą pokazano na **rys. 3** i **4**. W prezentowanym projekcie obsługa magistrali jest realizowana programowo. Procedury zapisu i odczytu w języku C przedstawione są na **list. 1** i **list. 2**.

Jak już wspomniałem, oprócz interfejsu równoległego można wykorzystać też interfejs szeregowy. Pozwala to połączyć S7600A z mikrokontrolerami za pomocą niewielkiej liczby linii, co ma ogromne znaczenie na przykład dla mikrokontrolerów podobnych do PIC16F84. Ten tryb jest wybierany przez wymuszenie na wejściu PSX poziomu niskiego. Linia SD6 magistrali danych jest wtedy wejściem sygnału zegarowego. Linia SD5 to wejście danych, natomiast SD7 wyjście danych (patrząc od strony S7600A). Kierunkiem przepływu danych steruje wejście WRITEX. Poziom wysoki na wejściu WRITEX oznacza odczyt danych, a poziom niski zapis danych. Przebiegi czasowe interfejsu szeregowego pokazano na **rys. 5** (cykl zapisu) i **6** (cykl odczytu).

Tab. 2. Przestrzeń adresowa Banku 0.

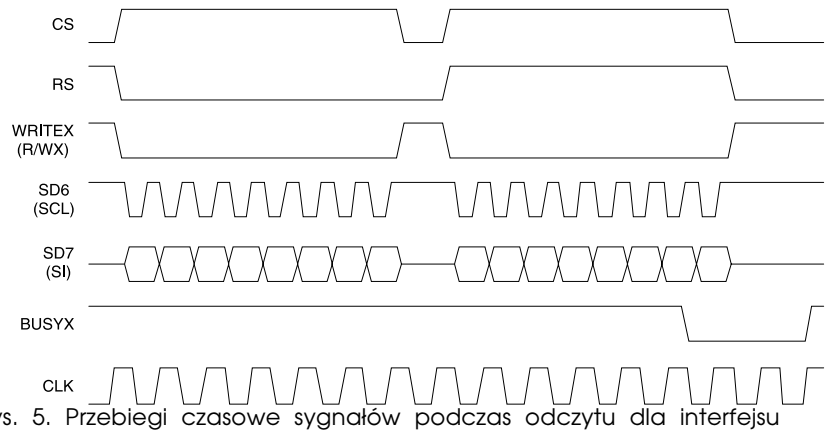
Adres	Rozmiar	Zawartość
0x0000-0x07ff	2k	Kieszeń 0 bufora odbioru
0x0800-0x0bff	1k	Kieszeń 0 bufora nadawczego
0xc000-0x0fff	1k	Dane bazowe TCP
0x1000-0x13ff	1k	Bufor IP

Tab. 3. Przestrzeń adresowa Banku 1.

Adres	Rozmiar	Zawartość
0x0000-0x07ff	2k	Kieszeń 1 bufora odbioru
0x0800-0x0bff	1k	Kieszeń 1 bufora nadawczego
0xc000-0x0fff	1k	Bufor PPP
0x1000-0x13ff	1k	Bufor PAP

W strukturze układu S7600A umieszczony jest kompletny port szeregowy UART. Tor odbiorczy zawiera 16-bajtowy bufor FIFO. Dane przesyłane asynchronicznie mają następujący format: 1 bit startu, 8 bitów informacyjnych i 1 bit stopu, bez bitu parzystości.

Zasadniczym blokiem układu jest jednak sprzętowy stos TCP/IP. Zawiera on moduły TCP/UDP, moduł IP, oraz moduł PPP. Z protokołami TCP/UDP/IP/ i PPP są związane 2 kieszenie umieszczone w wewnętrznej pamięci RAM. Cóż to takiego te kieszenie? Otóż są to obszary pamięci RAM, w których umieszcza się dane do przesyłania za pomocą protokołu TCP/IP. Oprócz kieszeni, w pamięci RAM podzielonej na banki po 5kB umieszczone są bufory pomocnicze protokołów TCP, IP, oraz PPP. Podział pamięci pokazano w **tab. 2** i **3**. Może się zdarzyć, że przy takim podziale pamięci



Rys. 5. Przebiegi czasowe sygnałów podczas odczytu dla interfejsu szeregowego.

Tab. 4. Rejestry układu S7600A.

Adres	Rejestr	Definicja bitu							
0x00	Revision	Numer wersji rdzenia S-7600A							
0x01	General Control	-	-	-	-	-	-	-	SW_RST
0x02	General Socket Location	0	0	0	0	0	0	S1	S0
0x04	Master Interrupt	-	-	-	-	-	PT_ INT	Link_ INT	Sock_ IN
0x08	Serial Port Config	S_DAV	DCD	Dsr Hwfc	CTS	RI	DTR	RTS	SCTL
0x09	Serial Port Int	PT_Int	-	-	-	-	-	-	-
0x0a	Serial Port Int Mask	PINT_ EN	DSINT _EN	-	-	-	-	-	-
0x0b	Serial Port Data	Rejestr danych portu szeregowego							
0x0c 0x0d	Baud Rate Div	Rejestry określające prędkość transmisji							
0x10-0x13	Our IP Address	Adres IP serwera dostępowego							
0x1c	Clock Div Low	Rejestr Clock Divider							
0x1d	Clock Div High								
0x20	Index	Rejestr indeksowy kieszeni							
0x21	TOS	Pole TOS							
0x22	Socked Config status Low	T0	Buff Emty	Buff Full	D_A /RST	-	Protocol Type		
0x23	Socked Status Mid	URG	RST	Term	Conu	Stan TCP			
0x24	Socked Activate	-	-	-	-	-	-	S1	S0
0x26	Socked Interrupt	-	-	-	-	-	-	I1	I0
0x28	Socked Data Avail	-	-	-	-	-	-	Dav1	Dav0
0x2a	Socked Interrupt Mask Low	T0 En	Buff_E En	Buff_ Full	DataA En	-	-	-	-
0x2b	Socked Interrupt Mask High	Urg_En	RST-En	Term_ En	ConU_ En	-	-	-	-
0x2c	Socked Interrupt Low	T0	Buff_ Empty	Buff_ Full	Data Avail	-	-	-	-
0x2d	Socked Interrupt High	URG	RST	Term	Conu	-	-	-	-
0x2e	Socked Data	8-bitowe dane kieszeni							
0x30	TCP data Send	Wpisanie dowolnej wartości rozpoczyna wysyłanie danych							
0x30-0x31	Buffer Out Lenght	Wielkość bufora wyjściowego (dla czytania tych rejestrów)							
0x32-0x33	Bufer In	Wielkość bufora wejściowego (dla czytania tych rejestrów)							
0x34-0x35	Urgent Data Pointer	Wskaźnik ważnych danych w buforze wejściowym/ wielkość datagramu UDP							
0x36-0x37	Their port	Numer portu docelowego							
0x38-0x39	Our port	Numer portu źródłowego							
0x3a	Socket Status High	-	-	-	-	-	-	-	Snd-bsy
0x3c-0x3f	Their IP address	Adres IP docelowy							
0x60	PPP Control Status	PPP_ Int	Con_ Val	Use_ PAP	T0_ Dis	PPP Int En	Kick	PPP_ En	PPP Up SRset
0x61	PPP Interrupt Code	Kod błędu PPP							
0x62	PPP Max Retry	-				Maks. liczba powtórzeń config request			
0x64	PPP String	Nazwa użytkownika i hasło							

mogą występować konflikty przy dostępie, tzn. 2 lub więcej modułów może chcieć w tym samym momencie odwoływać się do jednego wspólnego obszaru. Zadaniem arbitra dostępu do pamięci (**rys. 7**) jest właśnie bezkonfliktowe przydzielanie jej zasobów do poszczególnych modułów stosu.

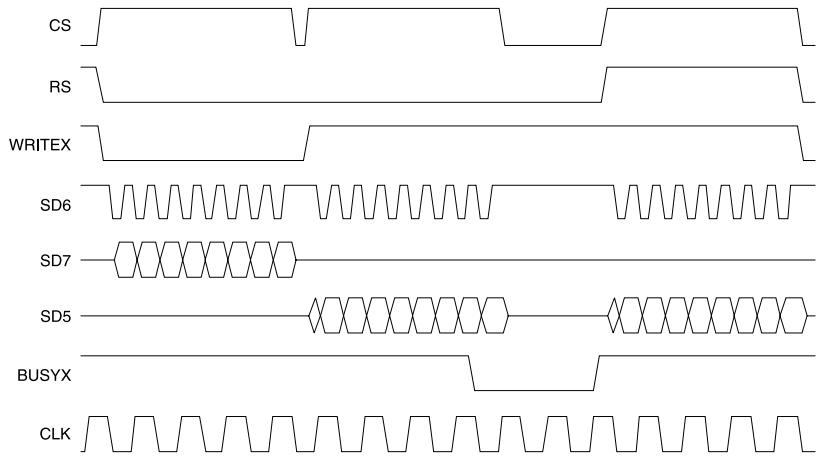
Układ S-7600A zawiera dwa 5-kilobajtowe banki pamięci (0 i 1), jak to pokazano w tab. 2 i 3.

Rejestry wewnętrzne S-7600A są podzielone na 3 grupy: globalne, bezpośrednie i indeksowe. Rejestry globalne zajmują przestrzeń adresową od adresu 0x00 do 0x1d, oraz od 0x60 do 0x6f. Pośrednie i bezpośrednie rejestry zajmują przestrzeń od adresu 0x20 do 0x3f. Użycie rejestrów indeksowych wymaga wcześniejszego zdefiniowania indeksu kieszeni. Zależnie od tej definicji, dane w rejestrach indeksowych dotyczą kieszeni 1 lub 2. W **tab. 4** pokazano zestawienie wszystkich rejestrów S7600A.

Układ S7600A jest przystosowany do fizycznego połączenia z Internetem za pośrednictwem modemu. Interfejs warstwy fizycznej wyposażony jest w związku z tym we wszystkie sygnały sterujące łącza RS232, potrzebne do prawidłowej współpracy z modemem. Sygnały te mają poziomy napięć standardu TTL, a jak wiadomo modemy wymagają poziomów zgodnych ze standardem RS232. Odpowiedni konwerter zbudowany jest w oparciu o układ U3 MAX3241. Zasilanie układu S7600A napięciem +3,3V wymusiło zastosowanie konwertera również zasilanego tym napięciem. Złącze Z2 jest 9-pinowym męskim złączem szufladowym. Sygnały na Z2 dołączone są do jego pinów

WYKAZ ELEMENTÓW

Rezystory
R1: 2,7MΩ
R2...R11: 3,3kΩ
Potencjometr 4,7kΩ
Kondensatory
C1...C5, C12, C13: 100nF
C6, C7: 33pF
C8: 2,2µF/16V
C9, C10: 33pF
C11: 1000µF/16V
C14, C15: 10µF/16V
Półprzewodniki
M1: 1A/100V
U1: S7600A
U2: 89C51 - zaprogramowany
U3: MAX3241
U4: CD4001
U5: CD4027
U6: MAX604
U7: 7805
Różne
X1: rezonator kwarcowy 12MHz
X2: rezonator kwarcowy 1MHz
Wyświetlacz alfanumeryczny 2x20 znaków
Z1 Złącze szufladowe 9-pinowe
Podstawka 40 DIL
SW1 przycisk typu switch
Płytka drukowana



Rys. 6. Przebiegi czasowe sygnałów podczas zapisu dla interfejsu szeregowego.

zgodnie ze standardem stosowanym w komputerach PC. Można bez problemu podłączyć do układu dowolny modem zewnętrzny za pomocą standardowego kabla używanego do połączenia modemu z komputerem.

Sygnał o częstotliwości 1MHz, wytwarzany w generatorze zbudowanym ze zlinearyzowanej bramki U4A i rezonatora X2, jest następnie dzielony przez 4 w dwu przerzutnikach U5A i U5B. Układy U4 i U5 są również zasilane napięciem +3,3V. Prostokątny przebieg o częstotliwości 250kHz i amplitudzie zbliżonej do napięcia zasilania podawany jest na wyprowadzenie 3 U1.

Mikrokontroler U2 też jest zasilany napięciem +3,3V. Do portu P2 dołączony jest wyświetlacz alfanumeryczny 2x20 znaków. Dość trudno jest znaleźć taki wyświetlacz zasilany obniżonym napięciem, dlatego zastosowano popularny wyświetlacz zasilany napięciem +5V. Linie portów mikrokontrolera zasilanego napięciem niższym niż +5V mogą być „podciągane“ do +5V bez szkody dla układu. Stabilizator U6 7805 dostarcza napięcia +5V, a układ U5 MAX604 napięcia +3,3V.

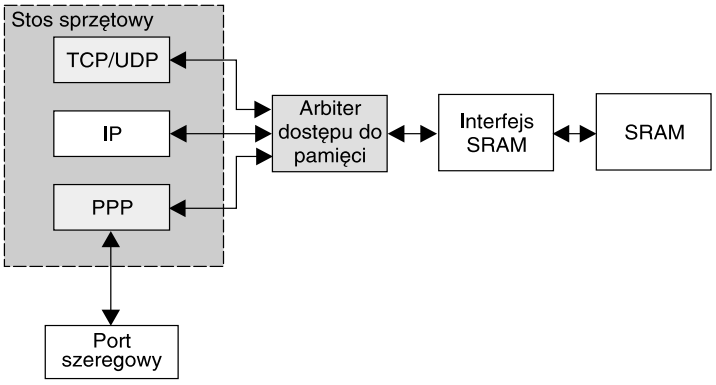
Montaż układu

Płytka drukowana interfejsu pokazana jest na **rys. 8**. Układ S7600A jest umieszczony w 48-pinowej obudowie typu QFP przystosowanej do montażu powierzchniowego. Niestety przylutowanie układu jest dość trudne. Odległość między nóżkami obudowy wynosi tylko 0,5mm! Przed lutowaniem układ należy przykleić do płytki drukowanej, najlepiej klejem typu Poxipol i dokładnie ustawić nóżki układu na polach lutowniczych, nie zapominając o prawidłowej ich kolejności. Po związaniu kleju można przystąpić do lutowania. Trzeba się wyposażyć w lutownicę z odpowiednio cienkim grotem i dobrą lupę, najlepiej na statywie. Montaż nie jest łatwy, ponieważ obudowy QFP zaprojektowano do montażu automatycznego, gdzie jest możliwe zachowanie wysokiej precyzji pozycjonowania i jakości lutowania.

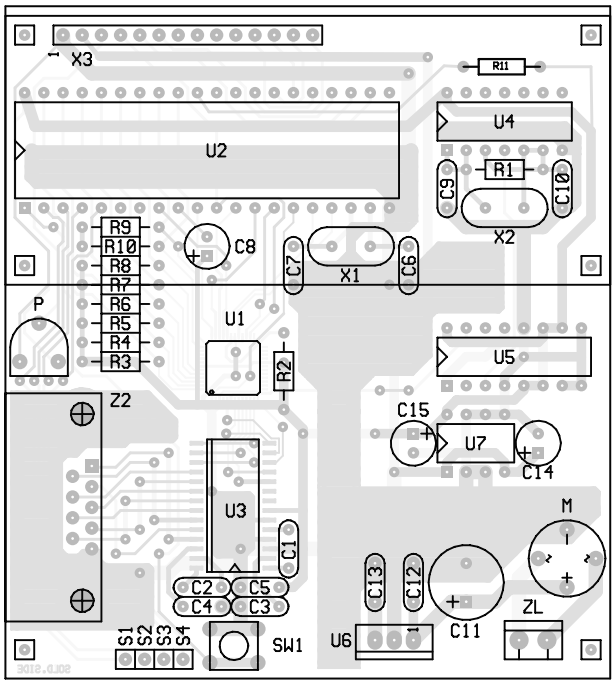
Po zakończeniu lutowania trzeba dokładnie sprawdzić, czy nóżki S7600A są dobrze przylutowane. Z doświadczenia wiem, że niektóre luty trzeba poprawiać kilka razy. Poprawki trzeba robić bardzo delikatnie, bo łatwo uszkodzić cienkie ścieżki lub wygiąć bardzo delikatne nóżki układu. Układ U3 jest produkowany również tylko w obudowie przystosowanej do montażu powierzchniowego, ale jego montaż nie nastręcza takich problemów jak to jest w przypadku S7600A. Rozstaw nóżek i ich grubość są zdecydowanie większe. Układ U3 również najlepiej jest wstępnie przykleić do płytki, zwracając uwagę na ustawienie nóżek na polach lutowniczych. Płytka jest tak zaprojektowana, że U3 trzeba przylutować na umownej stronie lutowania (pod spodem płytki). Montaż pozostałych elementów nie powinien sprawiać kłopotów.

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

Wzory płytek drukowanych w formacie PDF są dostępne w Internecie pod adresem: ***http://www.ep.com.pl/?pdf/marzec02.htm*** *oraz na płycie CD-EP03/2002B w katalogu* ***PCB****.*



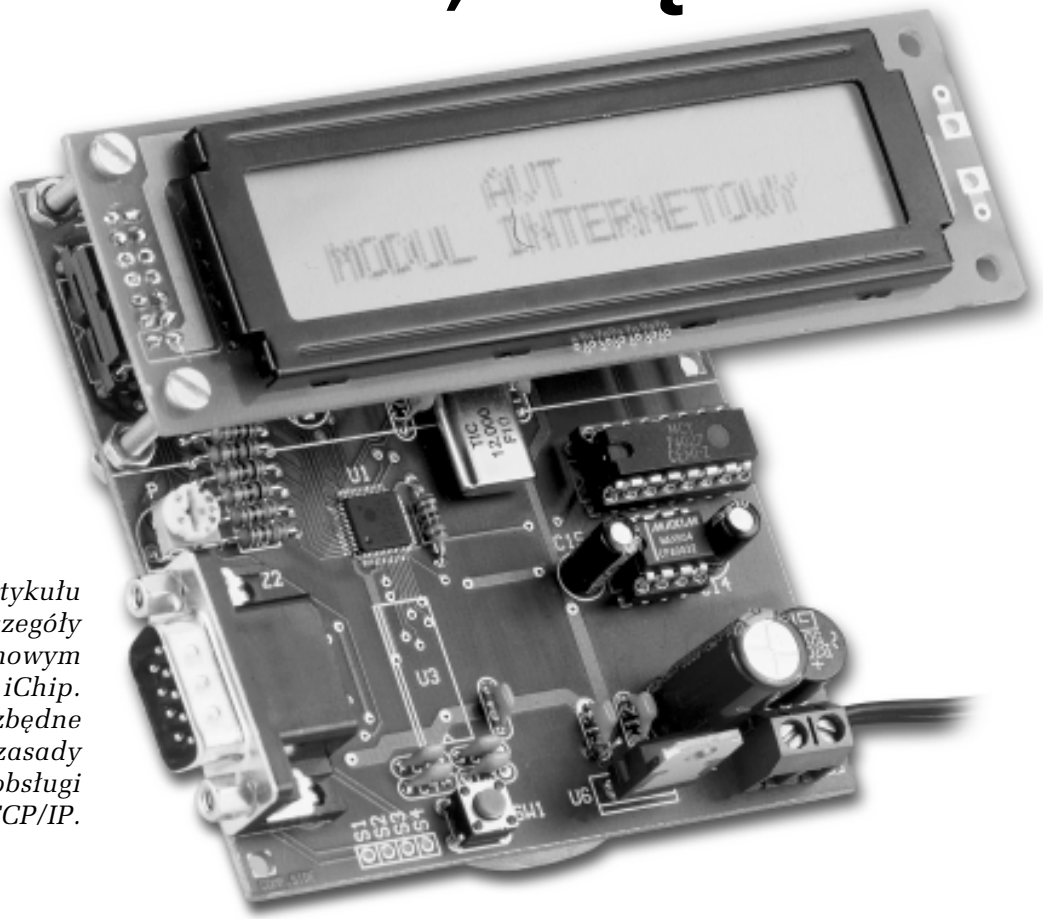
Rys. 7. Ilustracja kontrolowanego dostępu do pamięci.



Rys. 8. Rozmieszczenie elementów na płytce drukowanej.

Internetowy interfejs dla mikrokontrolera, część 2

AVT-5055



W drugiej części artykułu przedstawiamy szczegóły związane z programowym sterowaniem układu iChip. Informacje te są niezbędne do zrozumienia zasady działania i obsługi sprzętowego stosu TCP/IP.

Jak już wspomniałem do złącza Z2 trzeba dołączyć dowolny zewnętrzny modem, a układ zasilić napięciem stałym lub przemiennym o wartości ok. 8V. Po włączeniu zasilania mikrokontroler rozpoczyna procedurę inicjalizacji. Linie *cs* i *resetx* są zerowane, inicjowany jest licznik T1 (używany do odliczania opóźnień), oraz wyświetlacz LCD. Odblokowywany jest również system przerwań. Po wykonaniu tych czynności linia *resetx* jest ustawiana na „1" i do rejestrów *Baud Rate Divider* (adresy 0x1c...0x1d) wpisywane są wartości określające prędkość transmisji. Tą prędkość oblicza się według wzoru:

wartość programowa =
= [(fclk/pr_transmisji)]-1

gdzie: *fclk* to częstotliwości zegara S-7600A (uwaga: 0x0003 jest najmniejszą wartością, jaka może być wpisana do tych rejestrów!).

Przy obliczaniu wartości wpisywanych do rejestrów trzeba pamiętać, że dla niektórych prędkości błąd może być dość spory. Na przykład dla częstotliwości zegara 250kHz i prędkości transmisji 38400bd obliczona wartość do wpisania do rejestrów wyniesie: 250kHz/38,4kbd=6,5. Oczywiście możemy wpisać 6 albo 7. Dla wartości 6 rzeczywista prędkość wyniesie 250kHz/6 = =41,666kbd, a dla wartości 7 - 250kHz/7=35,714kbd. Widać, że rzeczywiste wartości różnią się znacznie od żądanych. Dla częstotliwości taktowania 250kHz optymalnym rozwiązaniem będzie prędkość transmisji 19200bd - 250kHz/19,2kbd=13,02. Rzeczywista prędkość transmisji wyniesie 250kHz/13=19,23kbd - jest to wartość, którą można zaakceptować. W naszym rozwiązaniu taka właśnie prędkość została wybrana. W momencie, kiedy płytka już

Tab. 5. Funkcje bitów rejestru Serial Port Configuration/Status (0x08)

Bit	Nazwa	Dostęp	Opis
7	S_DAV	R/W	*Serial Port Data Available* - czytany określa czy dane z portu szeregowego są dostępne. Powinno się tam wpisać 0.
6	DCD	R/W	*Carrier detect* - ten bit odzwierciedla stan linii DCD portu szeregowego. Jest on niezależny od ustawienia bitu SCTL.
5	DSR/ HWFC	R/W	*Data Send Ready/Hardware flow Control* - kiedy ten bit jest czytany, to określa bieżący stan linii DSR portu szeregowego. Kiedy jest zapisywany: 0 - *Hardware flow control* jest nieaktywny 1 - *Hardware flow control* jest aktywny
4	CTS	R	*Clear to send* (bit tylko do odczytu) - odzwierciedla stan bieżący linii CTS portu szeregowego.
3	RI	R	*Ring Indicator* - bit (tylko do odczytu) odzwierciedla stan linii RI portu szeregowego. Jest on niezależny od ustawienia bitu SCTL.
2	DTR	R/W	*Data Terminal Ready* - podczas odczytu określa bieżący stan linii DTR portu szeregowego. Mikrokontroler może kontrolować stan tego bitu poprzez wpisywanie.
1	RTS	R/W	*Request to send* - podczas odczytu określa bieżący stan linii RTS portu szeregowego. Mikrokontroler może kontrolować stan tego bitu poprzez zapisywanie.
0	SCTL	R/W	*Serial Port Control* - określa kto kontroluje port szeregowy. Wyzerowanie tego bitu (domyślnie) powoduje, że kontrolę przejmuje mikrokontroler. Ustawienie - sprzętowa kontrola przez stos sieciowy. 0 - MPU 1 - Kontrola sprzętowa

Tab. 6. Rejestr Serial Port Interrupt (0x09)

Bit	Nazwa	Dostęp	Opis
7	PT_INT	R	*Port Transport Interrupt* - ten bit sygnalizuje, kiedy przerwanie od portu szeregowego jest aktywne. Jest to zależne od stanu bitów PINT_EN i DSINT_EN w rejestrze Serial Port Interrupt Mask. Kiedy PINT_EN jest jedynką przerwanie wystąpi w momencie, kiedy dana jest dostępna w buforze FIFO portu (S_DAV w Serial Port Configuration/ Status jest też jedynką). Kiedy DSINT_EN jest jedynką przerwanie będzie aktywne kiedy CPU zapisuje do Serial Port Data Register, by transmitować daną. Jeżeli oba te bity są jedynkami, to przerwanie występuje w obu przypadkach.

Pozostałe bity nie są używane i po odczytaniu rejestru mają wartość 0.

Tab. 7. Rejestr Serial Port Interrupt Mask (0x0a)

Bit	Nazwa	Dostęp	Opis
7	PINT_EN	R/W	*Port Interrupt Enable* - zezwala na przerwanie od portu szeregowego
6	DSINT_ EN		*Data Send Interrupt Enable* - zezwala na przerwanie od wysyłania danych.

Pozostałe bity nie są używane i po odczytaniu rejestru mają wartość 0.

była gotowa udało mi się zdobyć rezonator o częstotliwości 2,4576MHz. Po podzieleniu przez cztery przez przerzutniki powstaje przebieg zegarowy o częstotliwości 614400Hz. Po wewnętrznym podzieleniu przez 32 uzyskujemy dokładnie 19200. Po podzieleniu przez 16 można uzyskać dokładnie 38400bd. Okazało się jednak, że przy częstotliwości przebiegu zegarowego 614,4kHz układ przestał prawidłowo pracować. Najprawdopodobniej spowodowane jest to zbyt dużą częstotliwością sygnałów jakie wystąpiły na magistrali S-7600A - większe częstotliwości zegara są przeznaczone dla szybkich procesorów. Pewnym rozwiązaniem tego problemu byłoby dodatkowe podzielenie częstotliwości zegara przez 2 za pomocą dodatkowego przerzutnika, ale jak wspomniałem płytka już była gotowa, więc pozostałem przy dotychczasowym rozwiązaniu, tym bardziej, że 19200bd jest prędkością zupełnie wystarczającą dla większości zastosowań.

W następnym kroku zapisywane są rejestry *Clock Divider* (0x1c...0x1d). Rejestry te służą do konfigurowania wewnętrznego zegara o częstotliwości 1kHz. Jest on używany do różnych funkcji czasowych w S-7600A: Licznik Podziału=(fclk/1kHz)-1, gdzie *fclk* to częstotliwości zegara S-7600A.

Dla f=250kHz 250kHz/1kHz-1= =249, co w zapisie szesnastkowym odpowiada wartości 0x00f9. Pod adres 0x1c wpisywana jest wartość 0xf9, a pod adres 0x1d wpisywane jest zero. Procedurę inicjalizacji kończy skonfigurowanie portu szeregowego poprzez wpisanie odpowiedniej wartości do rejestru *Serial Port Configuration/Status* (**tab. 5**).

Rejestr ten pozwala na sterowanie pracą portu szeregowego UART. Przez jego odczytanie można monitorować stany linii sterujących portu. Port szeregowy może być kontrolowany przez mikrokontroler lub przez sprzętowy wewnętrzny stos (bit SCTL). Wpisana w procedurze inicjalizacji wartość do rejestru *Serial Port Configuration/Status* powoduje, że kontrolę nad portem przejmuje mikrokontroler oraz włączony jest mechanizm *Hardware Flow Control* (bit DSR/HWFC = 1). Jeżeli mechanizm ten jest wyłączony, to dane są transmitowane niezależnie od poziomu sygnału *CTSX*. Jeżeli mikrokontroler kontroluje port to może on testować stan linii sterujących czytając rejestr *Serial Port Configuration*. Ma też kontrolę nad liniami DCD, DSR, DTR i RTS, a co za tym idzie nad wysyłaniem i odbieraniem danych.

Włączenie *Hardware Flow Control* powoduje, że zaczyna działać pełne sprzętowe potwierdzanie RTS/CTS. Kiedy port wykryje, że sygnał CTS przestał być aktywny, przestaje transmitować dane do momentu, aż ponownie stanie się on aktywny. W momencie kiedy bufor FIFO odbiornika portu jest zapełniony do połowy, to nieaktywny staje się sygnał RTS. Jest to sygnalizacja dla nadawcy, że ma zatrzymać nadawanie danych. Sygnał RTS ponownie staje się aktywny, jeżeli zostaną przeczytane dane z bufora wejściowego portu. Wpisanie danej do rejestru *Serial Port Data* kiedy port jest pod kontrolą mikrokontrolera powoduje, że dana ta zostanie wysłana z wcześniej zaprogramowaną prędkością transmisji. Zakończenie wysyłania sygnalizowane jest ustawieniem bitu *PT_INT* w rejestrze *Serial Port Interrupt* (**tab. 6**). Jednak żeby tak się stało, musi być ustawiony bit *DSINT_EN* oraz *PINT_EN* w rejestrze *Serial Port Interrupt Mask* (**tab. 7**).

Odczytywanie danych z portu odbywa się poprzez czytanie tego

Tab. 8. Rejestr PPP Control and Status (0x60)

Bit	Nazwa	Dostęp	Opis
7	PPP_Int	R/W	*Przerwanie PPP* - ten bit sygnalizuje, że zostało wyzwolone przerwanie od stosu PPP. Trzeba odczytać PPP interrupt code register dla stwierdzenia przyczyny. Zapisanie 1 do tego bitu kasuje przerwanie.
6	Con_Val	R/W	*Connection Valid* - ten bit sygnalizuje warstwie sieciowej, że połączenie „pod nią" jest poprawne 0 = połączenie niepoprawne 1 = połączenie poprawne
5	Use_ PAP	R/W	Ten bit odblokowuje autoryzację PAP w protokole PPP. Jeżeli jest odblokowana, to dostęp PAP jest dołączany po negocjacji autoryzacji PAP. Łańcuchy znakowe PAP są wprowadzane poprzez rejestr 0x64 0 = PAP zablokowany (domyślnie) 1 = PAP odblokowany
4	T0_Dis	R/W	*Timeouts Disabled* - ten bit blokuje blok PPP z timeoutami dla celów diagnostycznych. 0 = timeouts odblokowane (domyślnie) 1 = timeouts zablokowane
3	PPP_ Int_En	R/W	*PPP Interrupt Enable* - ten bit odblokowuje przerwanie PPP 0 = zablokowane (domyślnie) 1 = odblokowany
2	Kick	W	*PPP kick start* - kiedy wpisana jest 1 to bit ten wystartuje PPP, jeżeli zostało przerwane przez timeout. Bit ten jest samozerujący.
1	PPP_En	R/W	*PPP enable* - ten bit odblokowuje warstwę PPP i musi być ustawiony przed jakąkolwiek transmisją. 0 = PPP zablokowany (domyślnie) 1 = PPP odblokowany
0	PPP_UP/ SRst	R/W	Przy odczycie wskazuje, kiedy warstwa PPP ustanowi połączenie 0 = brak połączenia 1 = połączenie ustanowione Przy zapisie bit ten zeruje stos PPP. Jest samozerujący i nie potrzebuje zerowania podczas normalnej pracy 0 = normalna praca PPP 1 = reset PPP

samego rejestru *Serial Port Data*. Dane są gotowe do odczytu kiedy bit S_DAV w rejestrze *Serial Port Configuration/Status* jest jedynką.

Narzuca się pytanie: po co stosować tryb kontroli mikrokontrolera nad portem, skoro S-7600A może sam przejąć nad nim kontrolę? Otóż może to zrobić wówczas, gdy modem połączony do urządzenia i modem z drugiej strony ustalą między sobą połączenie. Przedtem jednak trzeba wysłać do modemu komendy sterujące, które zmuszą go do wybrania numeru dostępowego. Mikrokontroler musi odczytać z modemu informację o poprawnym zestawieniu połączenia modemowego, a może to zrobić tylko wtedy, gdy ma kontrolę nad portem. Modemy mogą wymagać specyficznej obsługi i jej sprzętowa realizacja może być trudna. Dla sprzętowego stosu protokołu PPP informacja, że połączenie w warstwie fizycznej jest poprawne oznacza, że można negocjować połączenie w warstwie sieciowej PPP. Właśnie wtedy S-7600A powinien przejąć kontrolę nad portem szeregowym.

Wróćmy jednak do modemu i sterowania jego pracą. Do komunikacji ze współcześnie stosowanymi modemami stosuje się standard potocznie nazywany „komendy AT". Nazwa ta utrwaliła się dlatego, że każda komenda wysyłana do modemu musi się zaczynać od znaków AT. Zestaw tych komend jest dość duży. W prezentowanym rozwiązaniu zastosowano dwie: ATV0 i ATDT. Pierwsza z komend przestawia modem na potwierdzenie komend w postaci numerycznej. Modem może odpowiadać po wykonaniu komendy takim właśnie kodem cyfrowym, lub ciągiem znaków ASCII będącym opisem wyniku działania komendy. Odpowiedź w postaci kodu wydaje się być prostsza w obsłudze i dlatego modem został przestawiony w taki właśnie tryb. Druga komenda powoduje, że modem wybiera numer wpisany jako parametr. Numer ten może być wybierany tonowo, lub impulsowo (ATDP). Komenda ta może mieć na przykład postać: ATDT 0202122. Wygląda znajomo - tak jest to numer dostępowy TPSA. Niestety, jak się dalej okaże, wysyłanie takiej komendy nie jest najlepszym pomysłem! Opis większości komend można znaleźć w instrukcjach obsługi modemów lub w Internecie.

Po wykonaniu komendy ATDT modem wybiera numer i łączy się z modemem połączonym z serwerem dostępowym. Jeżeli wysyłamy do modemu znaki komend AT z prędkością 19200bd, to modem próbuje nawiązać łączność z taką właśnie prędkością. Jeżeli nie jest to możliwe, zmniejszana jest prędkość transmisji, aż do uzyskania poprawnego połączenia. Po nawiązaniu łączności nasz modem odsyła kod określający prędkość transmisji wynegocjowaną w czasie połączenia i na taką prędkość należy przestawić S-7600A. W naszym rozwiązaniu zakładamy, że modem łączy się zawsze z prędkością 19200bd i jeżeli jest inaczej, to połączenie jest rozłączane.

Odebranie kodu po komendzie ATDT kończy fazę realizowania połączenia w warstwie fizycznej. W rejestrze *PPP Control and Status* o adresie 0x60 (**tab. 8**) można ustawić bit *Con_Val*, S-7600A może przejąć kontrolę nad portem i rozpoczyna się faza połączenia w warstwie sieciowej za pomocą stosu PPP.

Protokół PPP (*Point to Point Protocol*) jest tu używany do ustanowienia połączenia pomiędzy naszym urządzeniem a serwerem świadczącym usługi dostępowe. Warstwa PPP stanowi połączenie pomiędzy warstwą sieciową IP a warstwą fizyczną. Za pomocą PPP urządzenia na obu końcach łącza modemowego mogą wynegocjować sposób przesyłania ramek informacyjnych. Protokół dopuszcza negocjowanie wielu opcji i może się zdarzyć, że urządzenia nie będą mogły, mówiąc obrazowo, porozumieć się. Aby zapobiec takiej sytuacji wprowadzono pojęcie wartości domyślnych. Każda stacja MUSI przyjąć i potwierdzić opcje z tymi wartościami. Układ S-7600A podczas negocjacji połączenia PPP używa właśnie wartości domyślnych - powinien więc, przynajmniej teoretycznie, nawiązać połączenie z każdym serwerem.

Oprócz konfiguracji kanału transmisyjnego, PPP może być używany do potwierdzania autentyczności użytkownika w trakcie logowania się w serwerze (autoryzacja PAP - *Password Authenti-*

Tab. 9. Rejestr PAP String (0x64)

Bajt 0	0x05 Liczba znaków nazwy użytkownika
Bajt 1	Znak "t"
Bajt 2	Znak "o"
Bajt3	Znak "m"
Bajt 4	Znak "e"
Bajt 5	Znak "k"
Bajt 6	0x05 liczba znaków hasła
Bajt 7	znak "a"
Bajt 8	znak "l"
Bajt 9	znak "a"
Bajt 10	znak "m"
Bajt 11	znak "a"

cation Protocol). Podczas logowania przesyła się nazwę użytkownika oraz hasło. Wpisanie do bitu *PPP_En* jedynki powoduje włączenie PAP. Trzeba wtedy do rejestru *PAP String* (adres 0x64, **tab. 9**) wpisać kolejno znaki określające nazwę użytkownika i hasło. Pierwszy wpisywany do *PAP String* bajt określa liczbę znaków ASCII nazwy użytkownika. Następnie trzeba po kolei wpisywać te znaki. Po wpisaniu wszystkich wpisuje się bajt określający liczbę znaków hasła i, tak jak w przypadku nazwy użytkownika, znaki hasła.

Tab. 10. Rejestr PPP Interrupt Code (0x61)

Kod błędu	Definicja
0x00	Zarezerwowane
0x01	Niepowodzenie inicjalizacji fazy LCP protokołu PPP
0x02	Niepowodzenie negocjacji fazy NCP
0x03	Niespodziewane zakończenie LCP
0x04	Odebranie termination Request
0x05	Niepowodzenie negocjacji PAP

Po wpisaniu hasła i nazwy użytkownika pozostaje tylko wyzerowanie stosu PPP przez wpisanie do *PPP_UP/SRst* jedynki i odblokowanie warstwy PPP przez wpisanie jedynki do *PPP_En*.

Od tego momentu nie mamy żadnego wpływu na proces negocjowania połączenia PPP. Sprzętowy stos robi wszystko sam i pozostaje tylko czekać aż bit *PPP_UP/SRst* ustawi się na „1" i w ten sposób S-7600A zasygnalizuje poprawne połączenie.

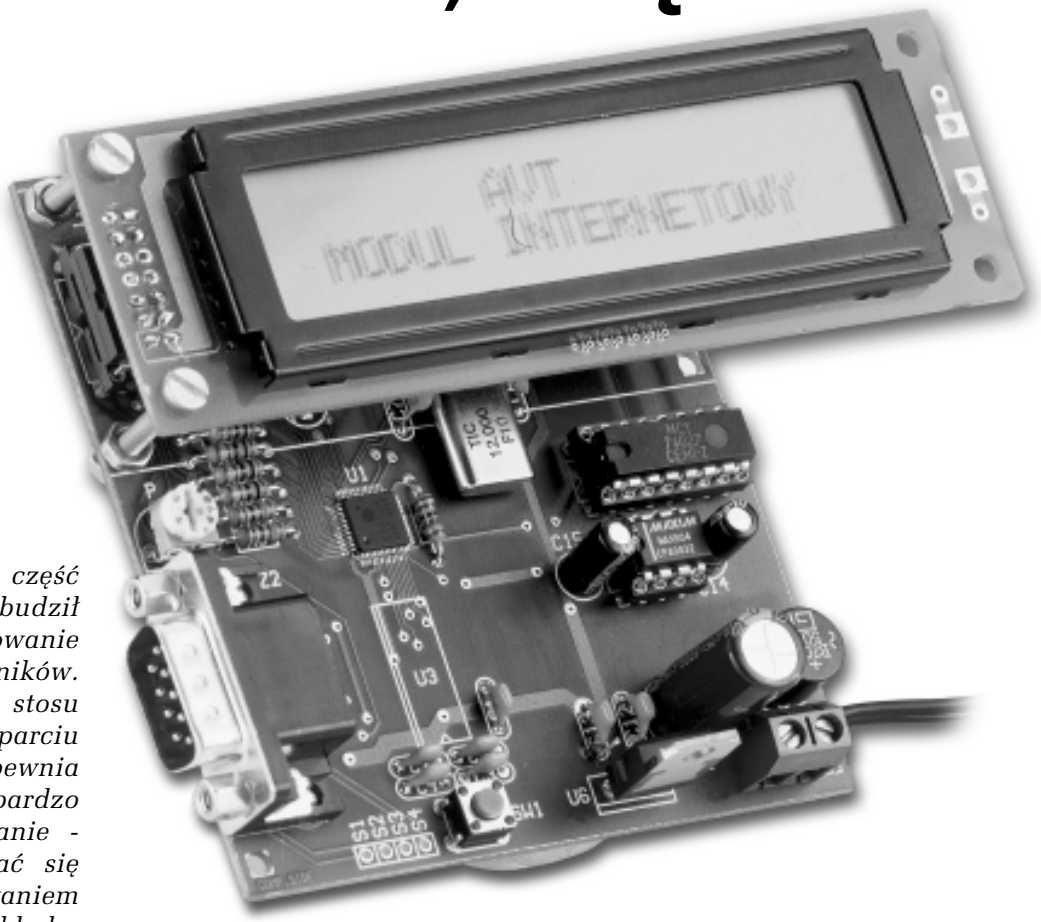
Gdy to połączenie nie może być zrealizowane, S-7600A może zgłaszać przerwanie przez wpisanie jedynki do rejestru *PPP Control and Status* na pozycji bitu *PPP_Int*. Trzeba wtedy przeczytać rejestr *PPP Interrupt Code* (adres 0x61) i określić przyczynę niepowodzenia (**tab. 10**).

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

Wzory płytek drukowanych w formacie PDF są dostępne w Internecie pod adresem: ***http://www.ep.com.pl/?pdf/kwiecien02.htm***.

Internetowy interfejs dla mikrokontrolera, część 3

AVT-5055



Jest to ostatnia część opisu projektu, który wzbudził ogromne zainteresowanie naszych Czytelników. Sprzętowa realizacja stosu TCP/IP, choćby w oparciu o układ S-7600A, zapewnia nowoczesne i bardzo praktyczne rozwiązanie - warto więc zapoznać się z programowaniem i konfiguracją tego układu.

Najbardziej dostępną i znaną metodą połączenia się z Internetem przez modem jest wykorzystanie numeru dostępowego TP S.A. Wystarczy wysłać do modemu komendę ATDT 0202122, a do *PAP String* sekwencję: 0x03 PPP0x03PPP i czekać na poprawne połączenie. Pierwsze próby, jakie robiłem nie dały jednak pozytywnego rezultatu. Powstawały zmiany w programie, kolejne próby i... niestety nic z tego. Zrobiłem wiele takich prób, straciłem mnóstwo czasu na modyfikację programu i wyczytywanie pakietów PPP wymienianych pomiędzy S-7600A a serwerami TP S.A. Okazało się, że nie jest możliwe ustanowienie połączenia pomiędzy S-7600A i serwerami TP S.A. Po zmianie na numer dostępowy firmy Polbox wszystko zaczęło działać od razu! Wszelkie próby dotarcia do kogoś z TP S.A., kto mógłby pomóc w wyjaśnieniu dlaczego tak jest skończyły się tak samo, jak próby łączenia się przez 0202122. Pozostawię to bez komentarza...

Jak już wspomniałem, wpisanie jedynki do *PPP_En* w *rejestrze PPP Control and Status* powoduje rozpoczęcie negocjowania połączenia przez sprzętowy stos PPP. Wcześniej trzeba wpisać jedynkę na pozycji bitu SCTL w rejestrze *Serial Port Configuration/Status*. Umożliwi to przejęcie kontroli nad portem szeregowym przez stos sprzętowy. Przy okazji trzeba też ustawić (wpisać jedynkę) bit DSR/HWFC i włączyć mechanizm *hardware flow control*.

Ze stosem PPP związane są jeszcze cztery rejestry *Our IP Address*. Pod adresem 0x10 znajduje się mniej znacząca część numeru IP, a pod 0x13 bardziej znacząca. Można tam wpisać adres IP serwera dostępowego lub nic nie wpisywać (domyślnie są

Tab. 11. Rejestr Config Status Low

Bit	Nazwa	Dostęp	Opis
7	TO	R	*TCP Timeout* - ten bit sygnalizuje, że wystąpił timeout w czasie ustanawiania połączenia TCP lub czekania na pakiet TCP po ustanowieniu połączenia. 0 = normalna praca 1 = nastąpił timeout
6	Buff_ Empty	R	Ten bit sygnalizuje, czy bufor danych wyjściowych jest pusty czy też nie. Bit jest jedynką, jeżeli bufor jest pusty. Jest zerowany, jeżeli kieszeń danych wyjściowych nie jest pusta i taka pozostaje. 1 = bufor pusty 0 = bufor nie jest pusty
5	Buff_Full	R	Ten bit sygnalizuje, czy przestrzeń jest dostępna do zapisu danych. Może również wyzwalać przerwanie, kiedy bufor jest pełny i bit *Buff_Full_En* w rejestrze *Socket Interrupt Mask Low* (0x2a) jest ustawiony. Rejestr *Data Register* nie powinien być zapisywany, kiedy Buff_Full=1. 0 = bufor dostępny 1 = bufor nie jest dostępny
4	Data_ Avail/RST	R/W	Wyzerowanie tego bitu ustawia wszystkie parametry kieszeni do wartości domyślnych. Jest samozerujący i nie potrzebuje zerowania dla właściwej operacji. Przed wyzerowaniem należy się upewnić, że bit *Snd_Bsy* w rejestrze *Socket Status High* (0x3A) jest wyzerowany. Przeczytany określa, czy są dostępne dane w kieszeni.
2:0	Protocol_ Type	R/W	Te bity są używane do ustawiania protokołu kieszeni. 010 = TCP Client Mode 101 = UDP Mode 110 = TCP Server Mode

tam wpisane same zera). W tym drugim przypadku serwer sam prześle swój adres i zostanie on wpisany do rejestrów *Our IP Address* (*floating IP address* negocjowany podczas sesji PPP).

Poprawne zakończenie negocjacji układ S-7600A sygnalizuje ustawiając bit *PPP_UP/SRst* w *PPP Control and Status*. Można wtedy odczytać rejestry *Our IP Address*.

Po ustawieniu *PPP_En* program sterujący czeka w pętli na ustawienie bitu *PPP_UP/SRst*. Jeżeli to nastąpi, to na wyświetlaczu (w górnym wierszu) pojawi się „PPP_ok.“, a w dolnym będą wyświetlone hexadecymalnie cztery bajty adresu IP.

W tym momencie S-7600A może rozpocząć nawiązywanie połączenia TCP ze stacją docelową. Do rejestrów *Their IP Address* (0x3c...0x3f) trzeba wpisać adres IP tej stacji. Przed dostępem do indeksowanych rejestrów kieszeni musi być zaprogramowany rejestr indeksowy o adresie 0x20. Kolejną czynnością jest wyzerowanie bitu *Data_avail/RST* (wyzerowanie kieszeni) w rejestrze *Config Status Low* (adres 0x22). Oprócz adresu IP potrzebne jest jeszcze określenie numerów portów źródła i przeznaczenia. Prezentowane urządzenie ma pracować jako klient poczty. W takim przypadku port źródła ma mieć losowo wybraną wartość z zakresu 1024...65535. Port przeznaczenia dla aplikacji używającej protokołu POP3 ma standardowy numer 110. Do rejestrów *Our Port registers* (0x38...0x39) wpisywana jest wartość 0x9200 (37376 dziesiętnie), natomiast do rejestrów *Their Port Registers* (0x37...0x37) jest wpisywana wartość 0x006e (110 dziesiętnie). Obie te wartości, tak jak i adres IP, muszą być ustawione przed aktywacją kieszeni. Po wpisaniu tych ustawień w rejestrze *Config Status Low* trzeba określić typ protokołu kieszeni na *TCP Client Mode*.

Teraz jest już wszystko gotowe i można rozpocząć połączenie TCP (aktywacji kieszeni) przez wpisanie dowolnej wartości do rejestru o adresie 0x30 (*Data Send and Buffer Length*). Jeżeli połączenie dojdzie do skutku, to w rejestrze *Socket Status Mid* (**tab. 11**) bit *ConU* zostanie ustawiony, a w polu *TCP State* powinna pojawić się wartość 2 (*established*).

Na wyświetlaczu w górnym wierszu pojawi się wtedy komunikat „Polaczenie TCP/IP ok“, natomiast w dolnym „port 110“. Jest to moment, w którym S7600A może wysyłać i odczytywać z serwera pocztowego dane. Dane są wysyłane w momencie wpisania do rejestru *Socket Data* (adres 0x2e). Odczytanie tego rejestru powoduje kolejne wczytywanie danych przychodzących z pamięci aktywnej kieszeni. Przez odczytanie *rejestrów Data Send and*

Bit	Nazwa	Dostęp	Opis
7	URG	R	Ten bit sygnalizuje przychodzące pilne dane. Wpisanie 1 na pozycji bitu URG w Socket Interrupt High Register (0x2d) zeruje ten bit 0 = nie ma pilnych danych 1 = są pilne dane
6	RST	R	Ten bit sygnalizuje, że kieszeń odebrała sygnał RST ze strony TCP 0 = nie odebrano sygnału RST 1 = odebrano RST
5	Term	R	Ten bit sygnalizuje, że kieszeń odłącza się od źródła i wyzwala przerwanie, jeżeli bit *Term_En* jest ustawiony w rejestrze *Socket Interrupt Mask High* (0x2b). Ustawienie maski przerwania nie powoduje braku przesyłania tego bitu. 0 = praca normalna 1 = odłączenie kieszeni od źródła ten bit ustawia się, kiedy S-7600A odbierze segment z flagą FIN. To oznacza, że zdalna strona żąda zamknięcia połączenia TCP.
4	ConU	R	Ten bit sygnalizuje, że kieszeń ustanowiła połaczenie z hostem. 0 = połączenie nieustanowione 1 = połączenie ustanowione
3:0	TCP State	R	Te bity sygnalizują bieżący stan TCP 0 = CLOSED 1 = SYN_SENT 2 = ESTABLISHED 3 = CLOSE_WAIT 4 = LAST_ACK 5 = FIN_WAIT1 6 = FIN_WAIT2 7 = CLOSING 8 = TIME_WAIT 9 = LISTEN A = SYN_RECVD

Buffer Length można określić wielkość bufora wejściowego.

Wymiana informacji może się odbywać tylko w ramach standardowego protokołu. Do odczytywania poczty stosuje się protokół POP3. Bez wdawania się w szczegóły, umożliwia on uwierzytelnienie użytkownika przez wysłanie nazwy i hasła, sprawdzenie liczby wiadomości i zajmowanej przez nie pamięci. Można też odczytać kolejne wiadomości i skasować wiadomość o konkretnym numerze. Po wysłaniu każdej komendy POP3, serwer odpowiada komunikatem, który zaczyna się „+ok." dla sytuacji kiedy komenda jest zaakceptowana i poprawnie wykonana lub „-err" jeżeli wystąpi błąd. Jeżeli wystąpi błąd, to program sterujący wyświetla odebrany komunikat i rozłącza połączenie przez wyzerowanie bitu *PPP_En* w rejestrze *PPP Control and Status*. W przypadku kiedy cała sekwencja odczytywania poczty jest poprawna, to można odczytać na wyświetlaczu całą przesłaną wiadomość. Zawiera ona oprócz przesyłanej treści - takiej, jaka pojawia się w oknie programu do odczytu poczty - szereg innych istotnych wiadomości normalnie niewidocznych (**list. 3**) - można tam znaleźć informację od kogo nadeszła wiadomość, dokładną datę i godzinę wysłania, a także dane identyfikacyjne serwera, z którego wiadomość została wysłana.

List. 3

```
+OK 979 octets
Return-Path: <tom50@poczta.onet.pl>
Received: from ghost3.onet.pl (ghost3.onet.pl [213.180.128.18])
by mach4.polbox.pl (8.10.2/8.10.2) with ESMTP id f9I6KKK9K052319
for <tomasz.jablonski@ep.com.pl>; Thu, 18 Oct 2001 20:14:09 +0200
Received: from pa24.zulavs.cvx.ppp.tpnet.pl ([213.77.198.24]:4100 "HELO
world")

by ghost3.onet.pl with SMTP id <S1078111AbRJRGON>;
Thu, 18 Oct 2001 20:14:13 +0200
Message-ID: <001d01c157bledd8b0080$0334521fjj9@world>
From: "Tomek" <tom50@poczta.onet.pl>
To: <tomasz.jablonski@ep.com.pl>
Subject: POP3 test
Date: Thu, 18 Oct 2001 20:13:19 +0200
MIME-Version: 1.0
Content-Type: text/plain;
charset="iso-8859-2"
Content-Transfer-Encoding: 7bit
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 4.72.3110.5
X-MimeOLE: Produced By Microsoft MimeOLE V4.72.3110.3
Status: RO

Witaj S7600A!
_
```

Oczywiście są tam też takie podstawowe informacje, jak temat i sama treść wiadomości. Ponieważ nawet najkrótsza wiadomość, to kilkaset znaków ASCII - w przedstawionym przypadku 979 znaków - to przeglądanie jej na ekranie wyświetlacza możliwe jest w sekwencjach zawierających 40 znaków. Pierwsze 40 znaków wyświetla się po odebraniu, a kolejne po przyciśnięciu przycisku SW1.

Podsumowanie

Przedstawiony tutaj projekt jest dość nietypowy. Nie jest to projekt zamkniętego urządzenia, ale raczej „szkieletu" służącego do opracowania własnych konkretnych aplikacji.

Starałem się przedstawić w artykule, w jak najbardziej przejrzysty sposób, jak można wykorzystać S-7600A w aplikacji internetowej. Program sterujący działa poprawnie w sytuacji, kiedy połączenie jest realizowane właściwie i nie występują żadne problemy. Program, który zawierałby procedury obsługi sytuacji krytycznych byłby zdecydowanie bardziej rozbudowany, a co za tym idzie dużo mniej zrozumiały. Celowo też wybrałem w warstwie aplikacji protokół POP3. Można było wybrać np. HTML i ze strony WWW sterować jakimś elementem - np. przekaźnikiem. Jednak w takim przypadku jednocześnie do sieci musiałyby być podłączone: nasz moduł i komputer z otwartą stroną WWW, a to z kolei mogłoby zniechęcić wielu potencjalnych eksperymentatorów do prób. W przypadku poczty można najpierw wysłać na swoją skrzynkę wiadomość, a potem odczytać ją za pomocą modułu. Podczas testów opisywany moduł łączył się w zasadzie bez problemu z serwerem dostępowym Polboxu. Zdarzały się problemy przy logowaniu na serwerze POP3 - serwer odpowiadał komunikatem o zajętości. Jednak w czasie niewielkiego natężenia ruchu w Internecie połączenia następowały szybko i bezproblemowo. Do modułu podłączony był standardowy modem Zoltrix FM366 za pomocą firmowego kabla RS232. Do połączenia się ze stacją w Internecie potrzebny jest jej adres IP. Dostawcy usług internetowych zmieniają adresy IP zachowując tą samą nazwę domeny. Jeżeli aplikacja wykorzystuje system DNS do wyszukiwania adresów IP, to wszystko jest w porządku. W naszym przypadku tak nie jest i ważne jest by znać aktualny adres IP serwera pocztowego. Przy znajomości datagramów protokółu IP można ten adres wyczytać podczas sprawdzania poczty za pomocą np. Outlook Express. Można w tym celu wykorzystać dowolny terminal znakowy. Odpowiedni jest na przykład *Terminal Emulation* z *Nortona Commandera*. Adres IP serwera dostępowego może być wyświetlany na ekranie modułu po nawiązaniu połączenia PPP. Odpowiedni fragment programu w pliku *tcp.c* (opublikowany na płycie CD-EP3/2002B) jest zapisany w postaci komentarza i wystarczy go odpowiednio zmodyfikować, aby taka informacja mogła być wyświetlana. W pliku *tcp.c* należy też wpisać nazwę i hasło swojej skrzynki pocztowej, wpisać numer telefonu i całość skompilować. Program został napisany dla kompilatora C firmy Keil. W przypadku innych kompilatorów należy go odpowiednio zmodyfikować.

Mimo wielu sceptycznych głosów wydaje się, że systemy ste-

rowania czy akwizycji danych oparte na mikrokontrolerach nieodwracalnie wchodzą w świat wielkiej sieci. Przed konstruktorami otwierają się olbrzymie możliwości przesyłania danych na duże odległości. Dzięki S7600A, i na pewno doskonalszym następcom, aplikacje internetowe stają się łatwe i tanie. Nawet przedstawione tutaj rozwiązanie może służyć na przykład do okresowej zmiany parametrów sterowania oddalonych od siebie sterowników. Wystarczy za pomocą e-mail'a wysłać zbiór tekstowy zawierający potrzebne dane. Nic nie stoi na przeszkodzie, aby zaimplementować protokół SMTP i wysyłać do serwera wiadomości o zaistniałych zdarzeniach lub jakiś inny protokół np. HTTP.

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

Uwaga!

Kody źródłowe do projektu internetowego interfejsu opublikowaliśmy na CD-EP3/2002B.

Wzory płytek drukowanych w formacie PDF są dostępne w Internecie pod adresem: ***http://www.ep.com.pl/?pdf/maj02.htm*** *oraz na płycie CD-EP05/2002B w katalogu* ***PCB****.*