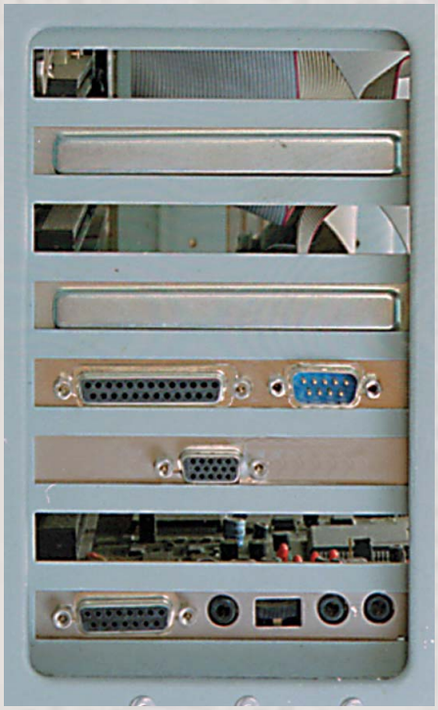
Wykorzystanie portów komputera PC

Port szeregowy — 2 część



Inne właściwości

Elektronik powinien znać podane powyżej ogólne zasady transmisji szeregowej złączem RS. W praktyce prawdopodobnie nie będzie jednak wykorzystywał tej wiedzy. Złącze to może, i często bywa wykorzystywane przez elektroników w nietypowy sposób. Należy więc poznać je bliżej.

W takich nietypowych zastosowaniach transmitowane sygnały nie mają struktury pokazanej na rysunku 5, a układ pracy nie wygląda tak, jak pokazuje rysunek 2. Wykorzystuje się za to dodatkowe linie łącza RS-232.

Każdy port szeregowy komputera PC, oprócz linii masy (GND), linii nadawania TxD oraz odbierania danych – RxD, ma jeszcze sześć dodatkowych linii oznaczanych DTR (Data Terminal Ready), RTS (Request to Send), CTS (Clear to Send), DSR (Data Set Ready), DCD (Data Carrier Detect) oraz RI (Ring Indicator).

Tabela 1

Wtyk DB-25	Wtyk DB-9	Sygnał	Funkcja
2	3	TxD	wyjście →
3	2	RxD	wejście ←
4	7	RTS	wyjście →
5	8	CTS	wejście ←
6	6	DSR	wejście ←
7	5	GND	Masa ⊥
8	1	DCD	wejście ←
20	4	DTR	wyjście →
22	9	RI	wejście ←

Wymienione linie są potrzebne na przykład przy współpracy komputera z modemem telefonicznym. **Rysunek 6** pokazuje jak zmieniają się stany poszczególnych linii, gdy modem przesyła informacje do komputera.

Podane skróty i angielskie określenia źródłowe mogą przyprawić o ból głowy. Na szczęście wcale nie trzeba rozumieć dokładnie, do czego miały służyć te wszystkie linie przy współpracy z modemem. Nie trzeba też szczegółowo analizować kolejności pojawiania się i znaczenia sygnałów na poszczególnych liniach – wystarczy wiedza, że łącze RS-232 zawiera linie pomocnicze.

Elektronik musi wiedzieć, że w komputerze linie oznaczane DTR i RTS mogą pełnić funkcje wyjścia, a linie oznaczane CTS, DSR, DCD oraz RI – mogą pełnić funkcję wejścia.

Trzeba też umieć sterować liniami DTR i RTS, oraz odczytywać stan linii CTS, DSR, DCD i RI. Jest to w sumie bardzo proste.

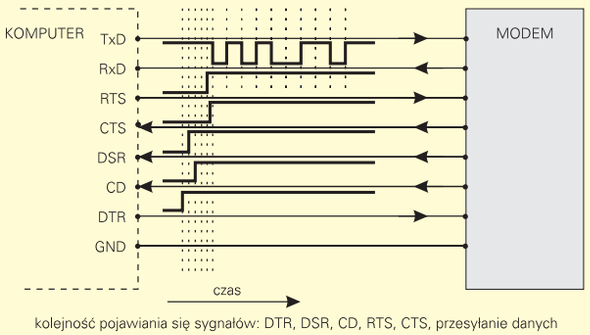
W tabeli 1 podano zwięzłu opis sygnałów portu szeregowego komputera i odpowiadające im numery końcówek złącz 9 i 25-stykowych. **Rysunek 7** pokazuje jak poszczególne linie podłączone są do szpilek gniazd.

W portach szeregowych spotyka się zarówno złącza 9-stykowe, jak i złącza 25-stykowe. Nie ma to większego znaczenia dla użytkownika, najwyżej na podstawie tabeli 1 lub rysunku 7 trzeba wykonać prostą przejściówkę składającą się z gniazda i wtyczki, łączącą szpilki o właściwych numerach.

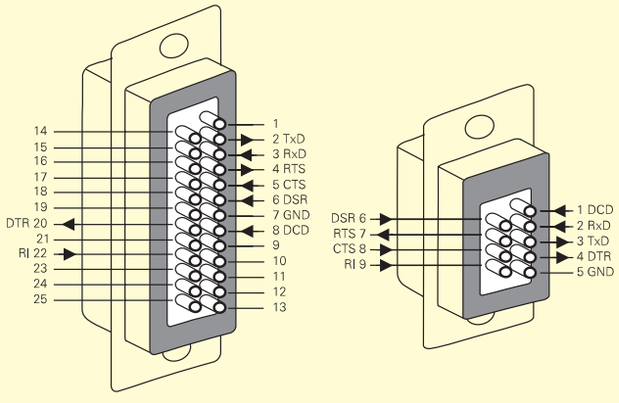
W komputerze porty szeregowe mają tak zwane złącza męskie, zawierające szpilki. Natomiast rysunek 7 pokazuje widok i numerację złącz żeńskich, czyli nasadek dołączanych do komputera. Praktyka interesuje właśnie to, jak okablować te nasadki dołączane do komputera. W wypadku jakichkolwiek wątpliwości co do numeracji nóżek złącz, należy odszukać cyferki wytłoczone na gniazdach i wtykach.

Poziomy napięć

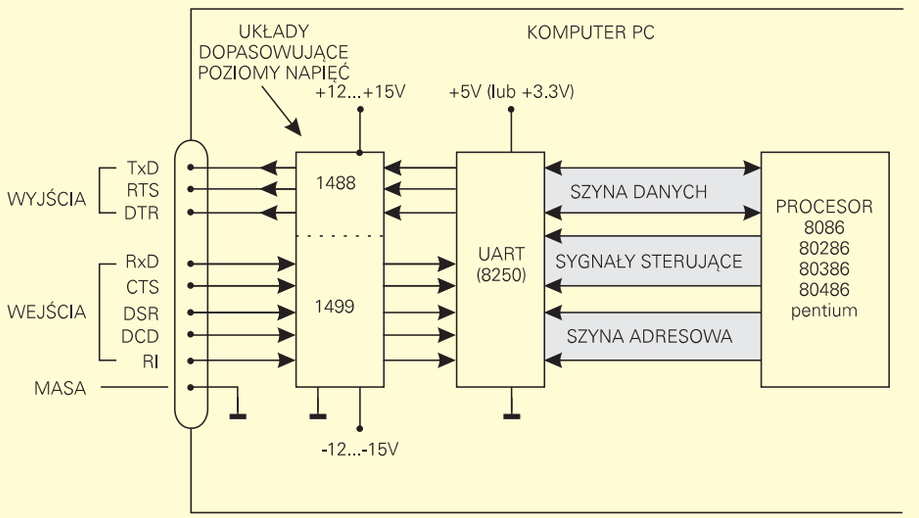
Jak wspomniano na początku artykułu, na wyjściach portu szeregowego występują napięcia rzędu ±10...15V. Takie poziomy napięć wywodzą się z epoki przedkomputerowej. W związku z przyjętymi wymaganiami na poziomy napięć, urządzenie zawierające port szeregowy musi zawierać obwody zasilania napięciem



Rys. 6. Stany poszczególnych linii przy przesyłaniu informacji z modemu do komputera.



Rys. 7. Podłączenie linii do szpilek złącz 25 i 9-pinowych.

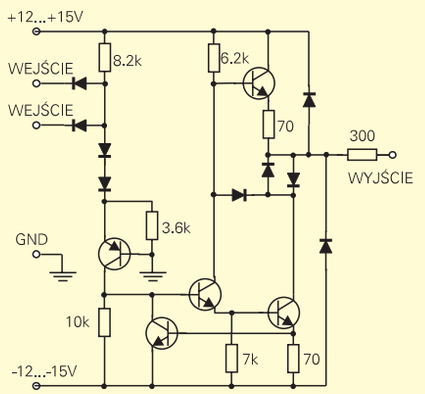


Rys. 8. Blokowy schemat portu szeregowego PC-ta.

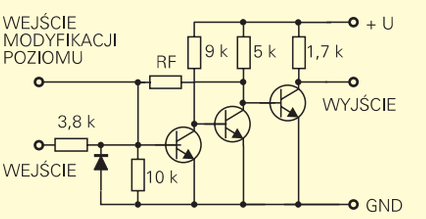
±10...15V albo też przetwornice, wytwarzające takie napięcia ze standardowego napięcia zasilania systemów logicznych, równego 5V lub 3,3V.

Schemat blokowy typowego portu szeregowego pokazany jest na **rysunku 8**. Procesor wpisuje do układu UART rozkazy sterujące, dane do przesłania i odczytuje odebrane dane. W komputerach PC wykorzystuje się kostkę UART 8250, 16450, albo stosuje się rozwiązania zgodne z nimi programowo.

Oprócz kostki UART konieczne są jeszcze układy dopasowujące poziomy napięć (0...+5V lub 0...+3,3V komputera, do napięć wymaganych w liniach łącza RS-232). Dawniej typowym układem odbiorczym była kostka o numerze 1488 i układem nadawczym – 1489. Te układy dopasowujące były zasilane napięciem symetrycznym ±10...15V. Schematy wewnętrzne jednego toru nadajnika 1488 i odbiornika 1489 pokazane są na **rysunku 9**. Obecnie bardzo wiele urządzeń przenośnych zasilanych jest z baterii, dlatego powszechnie stosuje się tam układy sprzęgające, które dodatkowo wyposażone są w system przetwornic, które z pojedynczego napięcia o wartości 5 lub nawet 3,3V wytwarzają napięcia symetryczne wymagane w standardzie RS-232. Nowoczesne kostki tego typu przedstawione zostaną niebawem w Klubie Konstruktorów.



Rys. 9a. Schemat wewnętrzny jednego toru nadajnika 1488.



Rys. 9b. Schemat wewnętrzny jednego toru odbiornika 1489.

Na **rysunku 10** pokazano dopuszczalne zakresy napięć na wyjściach i wejściach łącza RS dla stanów logicznych 0 i 1 według normy. Obok (rys 10c) podano, jak wejścia interpretują podawane na nie napięcia.

Z poziomami napięć na trzech wyjściach portu szeregowego jest trochę zamieszania, dlatego na rysunku 10 nie zaznaczono, jakie napięcia odpowiadają ligicznej 1, a jakie logicznemu 0. Mówi się, że informacja wysyłana przez linię TxD jest zanegowana. W rzeczywistości jest ona podwójnie negowana. Dla Czytelników EdW takie szczegóły nie są istotne. Najważniejszy jest fakt, że dwa urządzenia wyposażone w złacze RS-232 potrafią się porozumieć, o ile tylko jednakowo zaprogramowane zostaną parametry transmisji.

Elektronik, chcący wykorzystać port szeregowy powinien wiedzieć, że po włączeniu komputera, na wszystkich trzech wyjściach portu RS-232C występują napięcia ujemne.

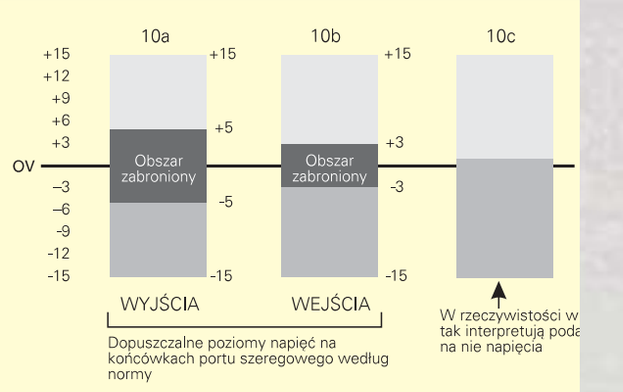
Wpisanie przez procesor do któregoś z wyjść portu (DTR lub RTS), logicznej jedynki (czyli ustawienie tego wyjścia), powoduje pojawienie się na odpowiedniej linii napięcia dodatniego o wartości +10...+15V. Ponowne wpisanie tam zera powoduje pojawienie się napięcia ujemnego o podobnej wartości.

Właśnie te napięcia można wykorzystać jako źródło zasilania dla dołączonych z zewnątrz układów. Ponieważ z wyjść można pobrać prąd rzędu kilku miliamperów, a współczesne kostki zużywają bardzo mało prądu, złącze RS może zasilać nawet dość rozbudowany układ.

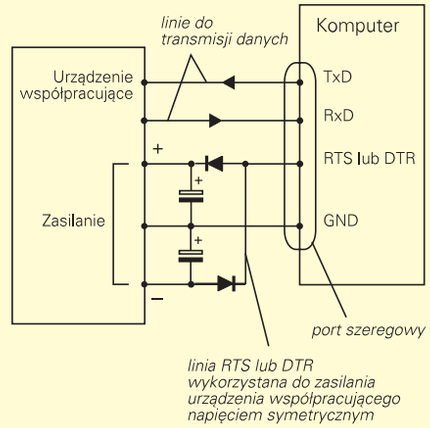
Przykładowo linie wyjściowe DTR i RTS (ustawione w przeciwnych stanach) mogą służyć jako źródło napięcia zasilającego, a linia RxD będzie pełnić funkcje wyjścia sygnałów.

W zasadzie już jedna linia wyjściowa może dostarczyć napięć symetrycznych. Wystarczy by komputer z odpowiednią częstotliwością ustawiał na niej na przemian stan wysoki i niski – napięcia symetryczne można uzyskać z pomocą dwóch diod i dwóch kondensatorów w układzie z **rysunku 11**.

Często do zasilania współpracującego urządzenia wystarczy napięcie o jednej biegunowości. Wykorzystuje się wtedy w bardzo prosty sposób jedną linię, i to linię, która jednocześnie transmituje dane.

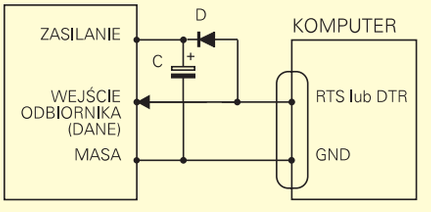


Rys. 10.



Rys. 11. Zasilanie bipolarne z portu RS.

Na **rysunku 12** pokazano sposób zasilania urządzenia i przesyłania do niego informacji za pomocą tylko dwóch przewodów.



Rys. 12. Zasilanie przez linię transmisyjną

Wiadomo, że z portu szeregowego na linię wysyłane są napięcia dodatnie i ujemne o znacznej wartości. Natomiast wejścia portu wcale nie muszą być sterowane takimi napięciami, jak pokazano na rysunku 10b. W zasadzie norma mówi, że sygnały przychodzące do wejścia powinny mieć napięcie (dodatnie lub ujemne) większe niż 3V. Ale w rzeczywistości układy odbiorcze (z nielicznymi wyjątkami) konstruuje się w ten sposób, że napięcia większe niż +1,25V traktowane są jako stan wysoki, a mniejsze niż +1V: jako stan niski. Wejścia wyposażone są ponadto w obwód histerezy, nie dopuszczającej do powstania przekłamań i drgań, także w zakresie napięć wejściowych +1...+1,25V. Pokazano to na rysunku 10c.

Jest to bardzo pożyteczne rozwiązanie. Dzięki niemu wejście portu szeregowego komputera może być sterowane sygnałami o poziomach TTL! Jest to bardzo ważna cecha, często wykorzystywana w praktyce. Port szeregowy komputera wysyła napięcia bipolarne, jak pokazano na rysunku 10a. Ale wejścia tego portu prawidłowo odczytają dane o poziomach TTL – to znaczy, że urządzenie współpracujące z komputerem wcale nie musi być wyposażone we wspomniane układy sprzęgające, zasilane napięciami symetrycznymi – wystarczy zasilanie pojedynczym napięciem w zakresie 3...15V. Takie właściwości wejść portu szeregowego otwierają przed amatorami duże możliwości i znakomicie upraszczają budowę urządzeń współpracujących.

Podane właśnie informacje są bardzo ważne dla każdego praktyka, bowiem pozwalają wykorzystać szeregowe porty komputera na wiele nietypowych, a bardzo ciekawych sposobów. W dalszej części artykułu podane będą propozycje prostych eksperymentów z portem szeregowym. Ale najpierw trzeba się nauczyć ustawiać i odczytywać stan poszczególnych linii portu.

Adresowanie

Z punktu widzenia procesora (a także z punktu widzenia programu, np. QBasica), obsługa portów polega na wpisywaniu i odczytywaniu bajtów informacji spod ściśle określonych adresów. Jest to bardzo proste: wpisujemy bajt pod określony adres i tym samym zmieniamy stany linii wyjściowych portu. Odczytujemy bajt spod pewnego adresu i otrzymujemy informacje o stanie linii wejściowych w momencie odczytu.

Elektronikowi-praktykowi nie są potrzebne wszystkie szczegóły na ten temat. Trzeba jednak rozumieć pewne podstawowe zagadnienie.

Z grubsza biorąc, w PC-cie mamy do czynienia z dwoma rodzajami adresów. Jak wiadomo, adresy są liczbami, wyrażanymi najczęściej w systemie szesnastkowym (dlatego liczby te poprzedzone są, lub zakończone, literką H-hexadecimal, a oprócz cyfr 0...9 zawierają znaki A...F).

Jedne adresy dotyczą współpracy procesora z pamięcią operacyjną (RAM). Zakres tych adresów sięga od zera do dziesiątków milionów – taki zakres jest potrzebny do obsługi dzisiejszych pamięci RAM o pojemności 32 lub 64 megabajtów.

Drugi rodzaj adresów dotyczy współpracy procesora z najróżniejszymi urządzeniami wejścia/wyjścia. Zakres tych adresów wynosi w zasadzie od zera do 65536 (szesnastkowo FFFFH), ale z tej liczby wykorzystuje się co najwyżej kilkaset adresów.

Mamy więc w PC-cie sytuację, że ten sam adres z zakresu 0...FFFFH może być adresem komórki w pamięci RAM, oraz adresem urządzenia wejścia/wyjścia. Komputer w sobie znany sposób radzi sobie z tymi podwójnymi adresami. Nas to w zasadzie może nie obchodzić, wystarczy, że poinformujemy prosesor, że ma on odczytać lub zapisać dane do urządzenia wejścia/wyjścia, a nie z/do pamięci RAM.

Jak się za chwilę okaże, jest to bardzo proste. Jest to tym bardziej łatwe, że w PC-cie dla poszczególnych urządzeń wejścia/wyjścia, a w tym także dla naszych portów, zarezerwowano pewne stałe adresy.

Mamy więc tak zwane adresy bazowe portów.

Adresy bazowe portów

Porty szeregowe COM1 i COM2 mają adresy bazowe (w zapisie szesnastkowym) 3F8 i 2F8, czyli w zapisie dziesiętnym 1016 i 760 (ewentualne porty COM3 i COM4 mają adresy szestnastkowe 3E8 i 2E8, co dziesiętnie daje 1000 i 744).

Pod każdym adresem znajduje się ośmiobitowy rejestr (coś w rodzaju komórki pamięci). Przy normalnej transmisji szeregowej (wg rysunków 2 i 5), pod te adresy wpisuje się (ośmiobitowe) dane do wysłania, i spod tych samych adresów odczytuje się dane odebrane z linii (z tego wynika, że w rzeczywistości pod tym jednym adresem bazowym są dwa rejestry: wyjściowy i wejściowy, ale dla nas nie jest to istotne).

Do zaadresowanego rejestru możemy więc wpisać (lub odczytać) dowolną liczbę z zakresu 0...255 (binarnie 0...11111111). To chyba jest oczywiste. A teraz uwaga! Bardzo często nie chodzi nam o liczbę, tylko o ustawienie w stan 1, lub wyzerownie poszczególnych bitów tego rejestru. Powiedzmy, że chcemy wpisać jedynkę do dwóch najstarszych bitów rejestru. To znaczy, że do tego rejestru musimy wpisać liczbę binarną 11000000. Liczba ta przedstawiona w zapisie dziesiętnym to 192. Z poziomu QBasica każemy więc wpisać pod określony adres liczbę 192 – tym samym wpiszemy dwie jedynki do najstarszych bitów rejestru. Analogicznie rozkaz wpisania liczby 7 wpisze jedynki do trzech najmłodszych bitów rejestru (bo liczba 7 to binarnie 00000111).

Jak wspomniano, w komputerze PC do obsługi portu szeregowego wykorzystuje się albo kostkę UART 8250, albo stosuje się rozwiązania zgodne programowo z tą kostką. W rzeczywistości do obsługi wszystkich funkcji portu szeregowego wykorzystuje się nie jeden (adres bazowy), ale kilka kolejnych adresów. Dla ułatwienia, zamiast dla każdego portu podawać wszystkie kolejne adresy, do adresu bazowego dodaje się tak zwany offset, czyli liczbę z zakresu 1...6.

Pokazuje to **rysunek 13**. Znów nie trzeba znać szczegółów. Ważna jest wiadomość, że aby wykorzystać wspomniane wcześniej dodatkowe linie portu szeregowego trzeba sięgnąć pod właściwe adresy.

Przykładowo jeśli pod adres szesnastkowy 3FC, (czyli 3F8+2, gdzie 2 to wspomniany offset) do najmłodszego bitu (D0)

wpiszemy jedynkę, czyli do rejestru wpiszemy liczbę binarną 00000001, to ustawimy linię DTR portu COM1. Gdy wpiszemy tam zero – powrócimy do stanu spoczynkowego linii zero. Tak samo możemy sterować linią RTS, wpisując pod ten sam adres do bitu D1 jedynkę lub zero. Jedynka na miejscu D1 to w zapisie binarnym 00000010, czyli liczba 2.

Uważny Czytelnik domyślił się samodzielnie, że można jednocześnie zmieniać stany obu wyjść DTR i RTS. Wpisanie pod adres 3FC liczby 3, czyli binarnie 00000011 ustawi oba wyjścia; wpisanie 00000010 wyzeruje linię DTR, pozostawiając ustawioną linię RTS, itd.

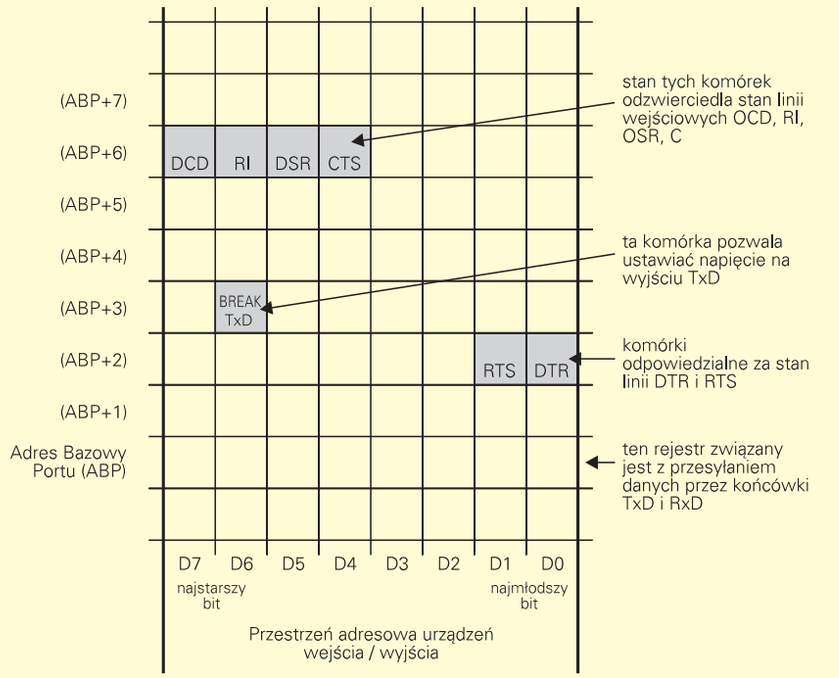
A więc rejestr o adresie (AdresBazowyPortu + 2) daje możliwość sterowania liniami DTR i RTS danego portu.

Jak wspomniano, wyjście TxD w zasadzie służy do szeregowego wysyłania 5...8 bitowych danych wpisywanych przez procesor do rejestru znajdującego się pod adresem bazowym (dla COM1 – 3F8H). W stanie spoczynku na linii TxD występuje napięcie ujemne. Ale w zastosowaniach nietypowych często po prostu ustawia się tę linię w jeden ze stanów. Ustawienie na wyjściu TxD napięcia dodatniego jest możliwe przez ustawienie siódmego bitu (D6), czyli wpisanie liczby dwójkowej 01000000, pod adresem (AdresBazowyPortu + 3). Późniejsze wpisanie tam zera znów zmieni napięcie wyjściowe TxD na ujemne itd...

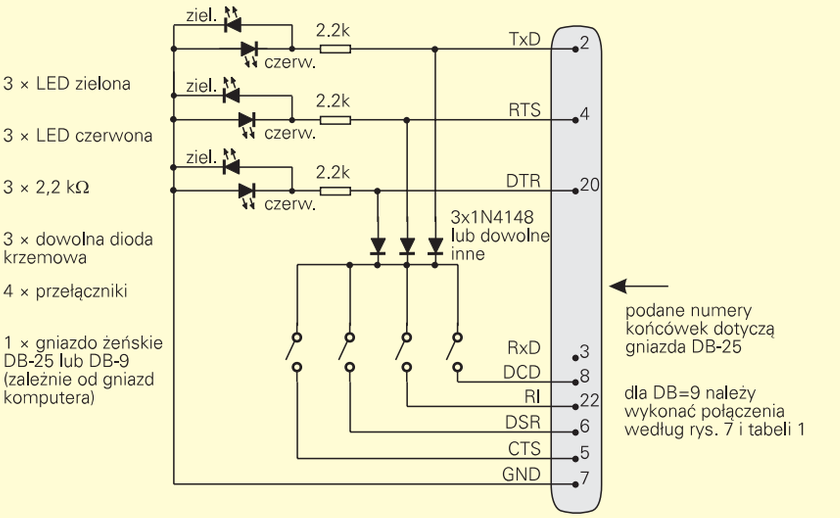
Naturalnie po wpisaniu tam jedynki wyjście TxD nie będzie wykorzystywane w typowy sposób pokazany na rysunkach 2 i 5.

Aby uzyskać informacje o stanie linii CTS, DSR, DCD i RI, wystarczy odczytać bajt spod adresu (AdresBazowy Portu + 6), czyli na przykład dla portu COM1 – będzie to adres (szesnastkowo) 3FE, czyli właśnie 3F8+6. Cztery najstarsze bity zawierają informacje o stanie tych wejść w momencie odczytu.

Podane właśnie wiadomości mogą w pierwszej chwili wydać się trudne, szczególnie dla początkujących. W rzeczywistości wykorzystanie ich jest niezmiernie proste, zwłaszcza za pomocą QBasica.



Rys. 13. Przestrzeń adresowa portu szeregowego.



Rys. 14. Schemat prostego układu do eksperymentów z portem szeregowym komputera PC.

Eksperymenty

Każdy posiadacz jakiegokolwiek komputera PC może od razu praktycznie wykorzystać podane właśnie wiadomości. Za pomocą łącza szeregowego PC-ta można sterować wielu urządzeniami i zbierać informacje od takich urządzeń. Do pierwszych fascynujących eksperymentów wystarczy elementarna znajomość języka QBasic. W poprzednich numerach EdW przedstawiono książki „Przygody z komputerem i bez komputera" oraz „QBasic nie tylko dla orłów". Książki te pomogą nawet zupełnie początkującym bezbolesne zapoznanie się z podstawami programowania. A wiadomości z niniejszego artykułu umożliwią różnorodne wykorzystanie zalet portu szeregowego.

Na początek warto zapoznać się praktycznie z działaniem poszczególnych linii portu.

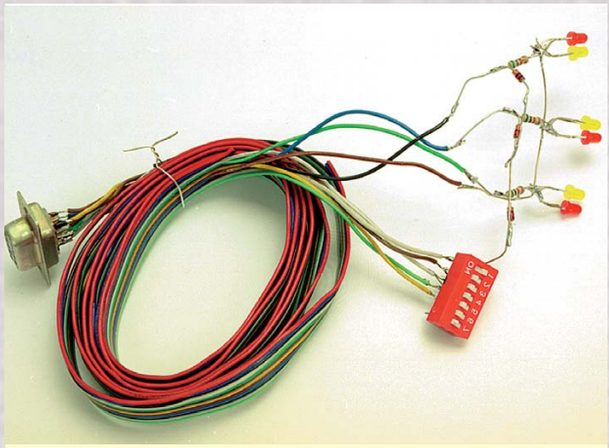
Na **rysunku 14** pokazano schemat najprostszego układu do eksperymentów z portem szeregowym.

Zmontowany układ widać na **fotografii 1**. Jeśli w komputerze występują złącza 9-pinowe, należy skorzystać z tabeli 1 i rysunku 7.

Programowanie

Dostęp do wspomnianych adresów z poziomu języka QBasic jest dziecinnie prosty. Najpierw należy podłączyć do wolnego portu szeregowego układzik o schemacie z **rysunku 14**. Jeden z portów szeregowych jest zwykle zajęty przez myszkę. Trzeba sprawdzić lub zapytać znajomego komputerowca, który port jest zajęty, a który wolny. Chodzi o ustalenie adresu bazowego wolnego portu. U piszącego te słowa port COM2 obsługuje myszkę, więc do eksperymentów posłuży port COM1 o adresie szesnastkowym 3F8.

Następnie trzeba uruchomić interpreter QBasic. Nie należy uruchomiać QBa-



Fot. 1.

sica spod Windows, bo próby się nie udadzą. Z poziomu DOSa wykonuje się to po prostu pisząc:

```
qbasic
```

i naciskając enter. Jeśli ten sposób nie uruchomi basica, należy podać ścieżkę dostępu, najczęściej będzie to:

```
c:\dos\qbasic
```

i nacisnąć enter, a za chwilę klawisz esc(ape), by usunąć z ekranu informację wstępną. Jeśli i to nie zadziała, trzeba poprosić o pomoc przy pierwszym uruchomieniu QBasica kogokolwiek, kto choć trochę zna się na komputerze.

Od chwili włączenia komputera, na liniach wyjściowych portu powinny wystąpić napięcia ujemne, czyli na początku świecić się będą diody czerwone.

Teraz można wpisać następujący programik:

```
ABP = &H3F8
OUT (ABP + 4), 3
```

i nacisnąć klawisz oznaczony F5 (lub wybrać z górnego menu Run, a potem Start).

W pierwszej linii programiku zadeklarowaliśmy adres bazowy naszego wolnego portu, do którego jest dołączony układ testowy. Ten adres to 3F8. Znak & wskazuje że jest to tak zwana stała liczbowa, a litera H przez adresem 3F8 – że jest to liczba szesnastkowa (Hexadecimal). Jeśli ktoś wykorzystuje port o innym adresie bazowym, przypisze stałej ABP aktualną wartość (zapewne ABP=&H2F8), reszta programu zostaje bez zmian.

W drugiej linii, poleceniem OUT, każemy do rejestru wyjściowego o adresie (AdresBazowyPortu+4) zapisać liczbę 3, czyli w zapisie binarnym 00000011. Tym samym ustawiamy bity odpowiedzialne za stan linii wyjściowych DTR i RTS (porównaj rysunek 13).

Po naciśnięciu klawisza F5 uruchomimy program – zmienimy stan linii DTR i RTS, zaświecą się dwie żółte diody.

Po naciśnięciu dowolnego klawisza wrócimy do ekranu QBasica.

Zmieńmy drugą linię programu:

```
ABP = &H3F8
OUT (ABP+4),0
```

Po naciśnięciu F5, znów zaświecą się wszystkie diody czerwone.

Po naciśnięciu jakiegokolwiek klawisza znów zmodyfikujemy program:

```
ABP = &H3F8
OUT (ABP+4),3
OUT (ABP+3),64
```

Druga linia programu ustawi wyjścia DTR i RTS, a trzecia linia, wpisując pod adres (AdresBazowyPortu+3) liczbę 64 czyli dwójkowo 01000000, zmieni stan linii TxD, czyli zaświeci trzecią żółtą diodę.

I to wszystko!

Jak się przekonałeś, sterowanie liniami RTS, DTR i TxD jest naprawdę dziecinnie łatwe.

Wpisz jeszcze krótki programik:

```
10 ABP = &H3F8
20 OUT (ABP + 4), 1
30 FOR N = 1 TO 1000: NEXT N
40 OUT (ABP + 4), 0
50 FOR N = 1 TO 1000: NEXT N
60 IF INKEY$ <> "" THEN END
70 GOTO 20
```

Diody sterowane przez linię DTR powinny zapalać się na przemian (z prędkością zależną od częstotliwości zegara taktującego twego komputera), aż do czasu naciśnięcia jakiegokolwiek klawisza. Zmień w linii 40 wpisywaną wartość z 0 na 2 i uruchom program – będą migać obie diody.

Aby odczytać stan linii wejściowych CTS, DSR, DCD i RI ustaw przynajmniej na jednym z wyjść (RTS, DTR lub TxD) napięcie dodatnie – przełączniki trzeba zasilić napięciem dodatnim. To napięcie zostanie podane przez diodę(y) na przełączniki. W zależności od stanu przełączników, na poszczególnych wejściach pojawi się albo napięcie dodatnie (przełącznik zwarty), albo napięcie masy (przełącznik zamknięty). Odczytaj zawartość rejestru spod adresu (AdresBazowyPortu+6) i zinterpretuj wyniki. Pomoże ci w tym programik zamieszczony u dołu strony.

W drugiej linii powyższego programu ustawiłeś dodatnie napięcie na liniach DTR i RTS. Kolejne linie sprawdzają stany poszczególnych bitów rejestru o adresie (AdresBazowyPortu+6) i wypisują na ekranie, czy na daną linię podano napięcie dodatnie, czy nie. Wykorzystałeś funkcję INP. Funkcja ta odczytała cały bajt informacji spod podanego adresu. Za pomocą iloczynu AND kolejno sprawdzane są stany czterech najstarszych bitów tego bajtu i w zależności od wyniku, wypisywany jest stosowny komunikat (pamiętaj, że np. 16 to binarnie 00010000, więc funkcja AND sprawdzi wartość bitu D4, wskazującego, jak podaje rysunek 13, stan lini CTS).

W ten prosty sposób nie możesz jednak odczytać stanu linii RxD. Niewielka strata, cztery linie wejściowe to też sporo.

Podane krótkie programy pokazują, w jak prosty sposób steruje się liniami wyjściowymi i odczytuje stan linii wejściowych. Otwiera ci to drogę do najróżniejszych zastosowań, oczywiście nie tylko do zapalania i gaszenia trzech diod LED.

Jeśli do tej pory nie splamiłeś się programowaniem i coś w podanych programikach nie jest dla ciebie jasne, powinieneś sięgnąć do książek o Basicu, przedstawionych w EdW 1/97 i 3/97 (kupony rabatowe jeszcze zachowują ważność).

Na życzenie Czytelników redakcja może przedstawić praktyczne przykłady wykorzystania portu szeregowego...

Napisz też do redakcji, jeśli jakiś szczegół nie jest dla Ciebie jasny.

(red)

```
ABP = &H3F8
OUT (ABP + 4), 3
IF (INP(ABP + 6) AND 16) = 16 THEN PRINT "CTS - plus" ELSE PRINT "CTS - minus lub masa"
IF (INP(ABP + 6) AND 32) = 32 THEN PRINT "DSR - plus" ELSE PRINT "DSR - minus lub masa"
IF (INP(ABP + 6) AND 64) = 64 THEN PRINT "RI - plus" ELSE PRINT "RI - minus lub masa"
IF (INP(ABP + 6) AND 128) = 128 THEN PRINT "DCD - plus" ELSE PRINT "DCD - minus lub masa"
END
```