RS 232C – praktyczne programowanie. Od Pascala i C++ do Delphi i Buildera. Wydanie II

Autor: Andrzej Daniluk
ISBN: 83-7197-990-8
Format: B5, stron: 412

Drugie wydanie książki „RS 232C – praktyczne programowanie. Od Pascala i C++ do Delphi i Buildera" to kompendium wiedzy adresowane do osób zajmujących się tworzeniem oprogramowania sterującego urządzeniami podłączanymi przez port szeregowy. Książka umożliwia zdobycie wiedzy niezbędnej, by przy użyciu narzędzi Delphi i Builder tworzyć w Pascalu i C++ własne programy do obsługi transmisji szeregowej.

W książce przedstawiono:

- Opis metod realizacji nowoczesnej transmisji asynchronicznej poprzez interfejs RS 232C
- Opis wszystkich, niezbędnych funkcji oraz struktur najczęściej wykorzystywanych do realizacji transmisji szeregowej oferowanych przez Win32 API
- Metody realizacji transmisji buforowanej oraz niebuforowanej
- Metody realizacji transmisji szeregowej z wykorzystaniem portów USB
- Kompletne przykłady oryginalnych algorytmów stosowanych przy obsłudze łącza szeregowego

Nowe wydanie zostało zaktualizowane i poprawione. Dodano nowe rozdziały opisujące tworzenie własnych komponentów obsługi portu szeregowego oraz transmisję szeregową z wykorzystaniem portu USB. Dokładny opis USB, standardu zdobywającego ostatnio ogromne uznanie, to kolejny powód, dla którego warto mieć tę książkę.

Dołączony do książki CD-ROM zawiera:

- Przykładowe programy napisane w C++ i Pascalu.
- Kompletne kody źródłowe aplikacji napisanych w C++Builderze 5.
- Kody źródłowe aplikacji napisanych w Delphi 6.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

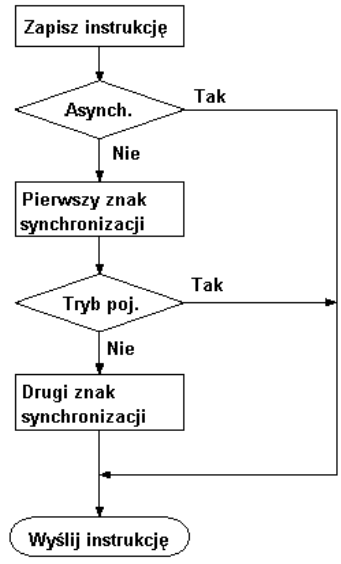
Rozdział 2.

Nowoczesna transmisja asynchroniczna oraz standard RS 232C

Podstawową wersję RS 232 (ang. *Recommended Standard*) wprowadzono w 1962 roku w USA. Początkowo standard ten miał służyć jedynie do obsługi modemów. Od tego czasu był poddawany kilkakrotnej aktualizacji celem bardziej optymalnego dostosowania do potrzeb szeregowej transmisji danych. Największą popularność zdobyła wersja RS 232C wprowadzona w 1969 roku, zaś oficjalnie do rangi standardu została podniesiona w roku 1986. RS 232C jest powszechnie stosowanym i akceptowanym standardem dla szeregowej wymiany danych cyfrowych pomiędzy urządzeniem DTE (ang. *Data Terminal Equipment*), obecnie utożsamianym z komputerem, a DCE (ang. *Data Communication Equipment*) — urządzeniem zewnętrznym (w oryginale — modemem). W sposób jednoznaczny definiuje on parametry elektryczne, mechaniczne i logiczne łącza szeregowego. Oficjalna jego nazwa brzmi *Interface Between Data Terminal and Data Circuit Termination Equipment Employing Serial Binary Data Interchange*. RS 232C stosowany bywa wszędzie tam, gdzie mniej istotną rolę odgrywa przepustowość łącza, natomiast ważna jest niezawodność i prostota obsługi protokołu komunikacyjnego.

Ze względu na prostotę realizacji szeregowej transmisji asynchronicznej większość obecnych urządzeń posługuje się właśnie takim trybem przesyłania danych. Różnice w sposobie realizacji transmisji synchronicznej i asynchronicznej schematycznie przedstawione są na rysunku 2.1.

Komputery osobiste wyposażone są w łącza szeregowe przystosowane do transmisji asynchronicznej, tzn. komputer i urządzenie muszą pracować z jednakową, wcześniej uzgodnioną prędkością oraz wykorzystywać taką samą strukturę znaków. Transmisja taka może być realizowana w trybie bez potwierdzenia odbioru lub z potwierdzeniem odbioru. Drugi sposób zapewnia nam możliwość kontrolowania poprawności wysyłanych-odbieranych danych. Dane przesyłane są w postaci tzw. ramki (ang. *frame*), która jest najmniejszą porcją możliwej do przesłania informacji. Bity przesyłane są kolejno.



Rysunek 2.1. *Różnica pomiędzy sposobami realizacji transmisji synchronicznej i asynchronicznej*

Do kodowania znaków stosuje się najczęściej kod ASCII (*American Standard Code of Information Interchange*). Początkowo stosowano 128 znaków zapamiętywanych na 7 bitach. W tym przypadku pierwszy bit danych — bit nr 0 poprzedzony był znacznikiem początku ramki — bitem startu. Ósmy bit (bit nr 7) służył do kontroli parzystości. Następnie przesyłany był znacznik końca ramki — jeden lub dwa bity stopu. Wraz z pojawieniem się strony kodowej ASCII o 256 znakach, pierwsze 32 znaki z przedziału 0 – 31 oraz znak 127 zaczęto rezerwować na potrzeby transmisji danych lub jako znaki sterujące dla urządzeń zewnętrznych. Obecnie zbiór ASCII jest podzestawem Unicode zawierającego 65 536 znaków, który używany jest do reprezentowania znaków większości języków świata. W tabeli 2.1 przedstawiono używane obecnie w transmisji szeregowej znaki sterujące. W dalszej części książki będziemy z nich często korzystać.

Obecnie ramka może zawierać od 5 do 8 bitów danych (jednak większość spotykanych urządzeń posługuje się słowem 7- lub 8-bitowym) poprzedzonych bitem startu oraz zakończonych bitem parzystości i jednym lub więcej bitami stopu. Przed rozpoczęciem transmisji bit startu przyjmuje zawsze wartość 0, zaznaczając wyraźnie moment początkowy. Odwrotność czasu trwania transmisji jednego bitu określa szybkość przesyłu w bitach na sekundę. Korzystając z funkcji BIOS-u, możemy uzyskać transmisję w granicach od 110 do 9600 b/s. Przekonamy się, że w Windows może być ona znacznie, znaczne większa. Powodem tych rozbieżności są pewne różnice w sposobie inicjalizacji procedur obsługujących łącze szeregowe stosowane w DOS i Windows. Możliwa do uzyskania szybkość transmisji zależy przede wszystkim od typu układu scalonego UART, w jaki zaopatrzona jest nasza płyta główna. Niezależnie od tego, że przetwarza on dane z postaci szeregowej na równoległą i odwrotnie, to obsługuje również sygnały sterujące interfejsu RS 232C. Czytelników zainteresowanych budową i możliwością programowania takich układów odsyłam do znakomitej książki Piotra Metzgera *Anatomia PC*, wyd. VII. Helion (2002).

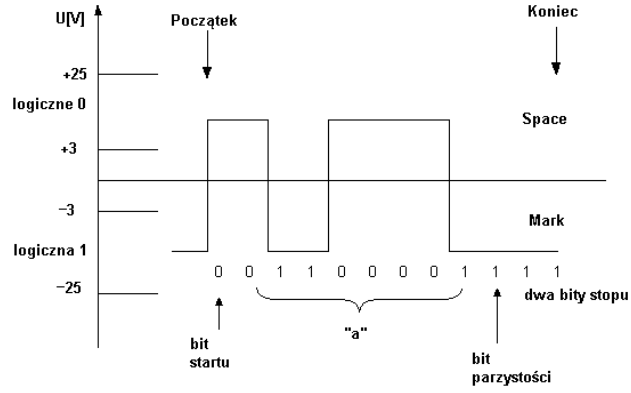
Tabela 2.1. *Pierwsze 32 znaki strony kodowej ASCII*

Liczba dzies.	Liczba heksad.	Znak	Nazwa angielska	Znaczenie
0	00	NULL	Null	Znak pusty
1	01	☺ SOH	Start of Heading	Początek nagłówka zawierającego adres lub polecenie
2	02	☻ STX	Start of Text	Początek tekstu
3	03	♥ ETX	End of Text	Koniec tekstu
4	04	♦ EOT	End of Transmission	Koniec transmisji
5	05	♣ ENQ	Enquiry	Zapytanie
6	06	♠ ACK	Acknowledge	Potwierdzenie
7	07	• BEL	Bell	Dzwonek
8	08	◘ BS	Backspace	Usuwanie poprzedniego znaku
9	09	○ HT	Horizontal Tabulation	Pozioma tabulacja
10	0A	◙ LF	Line Feed	Przejście kursora do następnego wiersza
11	0B	♂ VT	Vertical Tabulation	Pionowa tabulacja
12	0C	♀ FF	Form Feed, page eject	Przesuw strony
13	0D	♪ CR	Carriage Return	Powrót kursora do początkowej pozycji w tym samym wierszu
14	0E	♫ SO	Shift Out alternate character set	Następne znaki nie będą interpretowane jako znaki ASCII
15	0F	☼ SI	Shift In, resume default character set	Powrót do domyślnej strony kodowej ASCII
16	10	► DLE	Data Link Escape	Zmiana znaczenia następnego znaku, który będzie interpretowany jako kombinacja bitów sterujących danym urządzeniem
17	11	◄ DC1	Device Control 1	XON kontrola przepływu danych
18	12	↕ DC2	Device Control 2	Sterowanie urządzeniem
19	13	‼ DC3	Device Control 3	XOFF kontrola przepływu danych
20	14	¶ DC4	Device Control 4	Sterowanie urządzeniem
21	15	§ NAK	Negative Acknowledge	Meldunek błędu
22	16	▬ SYN	Synchronous Idle	Znak synchronizacyjny
23	17	↨ ETB	End of Transmission Block	Koniec transmisji bloku danych
24	18	↑ CAN	Cancel	Anulowanie danych
25	19	↓ EM	End of Medium	Koniec zapisu danych
26	1A	→ SUB	Substitue Character	Zastąpienie znaku, znak wstawiony w odbiorniku w miejsce błędnie odebranego, np. z błędem parzystości
27	1B	← ESC	Escape	Rezygnacja bez potwierdzenia
28	1C	∟ FS	File Separator	Separator plików
29	1D	↔ GS	Group Separator	Separator grup
30	1E	▲ RS	Record Separator	Separator zapisu
31	1F	▼ US	Unit Separator	Separator jednostek
127	7F	⌂ DEL	Delete	Unieważnienie znaku

Bit kontroli parzystości przesyłany za ostatnim bitem danych jest jedną z metod monitorowania poprawności transmitowanych danych. Z reguły przyjmuje dwie wartości: 0 lub 1. Ilość jedynek w polu danych może być uzupełniana do liczby parzystej (*evenparity*) lub nieparzystej (*oddparity*). Bit parzystości może być stale równy 1 (*markparity*), stale równy 0 (*space*) lub może być nieustawiony (*noparity*). Bity stopu zawsze oznaczają koniec ramki. Może wówczas nastąpić transmisja kolejnej „paczki" danych.

RS 232C jest interfejsem cyfrowym, zatem jego poziomom logicznym (0-1) należy przypisać określone przedziały napięć zarówno ujemnych, jak i dodatnich, które są z reguły nieco wyższe od stosowanych w komputerze. Pozwala to w dużym stopniu uniezależnić sygnał na wejściu interfejsu od przypadkowych zakłóceń. Dla sygnałów sterujących i sygnałów współpracy logicznej wartości 1 odpowiada przedział od +3 do +25 V, tzw. stan aktywny, wysoki, włączony lub „ON". Logicznemu 0 odpowiada przedział od –3 do –25 V, jest to stan nieaktywny, niski, wyłączony lub „OFF". Dla linii przesyłania danych logicznej wartości 1 (tzw. „*Mark*") opowiada przedział napięć od –3 do –25 V, zaś logicznemu zeru (tzw. „*Space*") przedział od +3 V do +25 V. Widzimy więc, że sygnały sterowania i współpracy są aktywne w stanie wysokim, zaś sygnały danych — w stanie niskim (*Mark*). Na rysunku 2.2 pokazano przebieg czasowy przykładowej ramki, ukazujący wysłanie jednej litery „a" reprezentowanej na ośmiu bitach — dziesiętnie 97, binarnie 01100001. Bit parzystości został ustawiony jako *markparity*, zastosowano też dwa bity stopu.

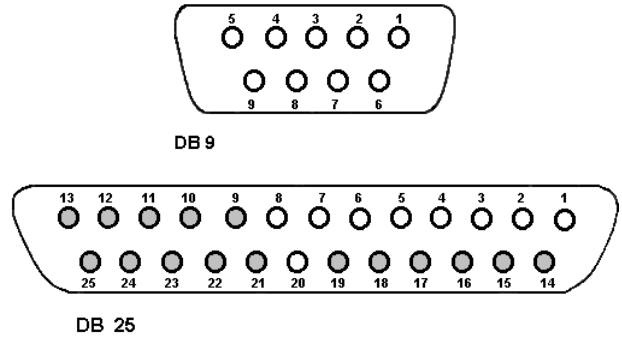
Rysunek 2.2. *Czasowy przebieg ramki na linii przesyłania danych przy wysłaniu litery „a"*



Łącze w trakcie ciszy utrzymywane jest stanie logicznej wartości 1. Transmisja rozpoczyna się od bitu startu, który zawsze przyjmuje wartość logicznego 0. Po nim następuje transmisja ośmiu bitów reprezentujących znak. Później jest bit parzystości, potem dwa bity stopu zamykające ramkę. Bitowi stopu odpowiada stan niski. Po nim łącze wraca do stanu ciszy. Jeden lub dwa bity stopu stosowane są po to, by odbiornik i nadajnik mogły dokonać wzajemnej synchronizacji przed transmisją kolejnej ramki danych. W praktyce układy nadajników zasilane są napięciem ±12 V, dając amplitudę sygnałów ±8 V. W tej sytuacji bitom wartości 0 oraz 1 transmitowanego bajta odpowiadają napięcia odpowiednio +12 V oraz –12 V[1].

[1] Zgodnie z zaleceniami protokołu V.28 CCITT (Międzynarodowy Komitet Doradczy ds. Telefonii i Telegrafii) logicznemu zeru powinien odpowiadać potencjał dodatni +3 V…+15 V, zaś logicznej jedynce potencjał ujemny –3 V…–15 V.

Standardową linię interfejsu RS 232C stanowi 25-żyłowy przewód, przy czym większość z tych linii wykorzystuje się dla potrzeb transmisji synchronicznej. W standardzie IBM wykorzystuje się jedynie 9 sygnałów, które są wystarczające do zrealizowania transmisji asynchronicznej. W komputerach PC używano początkowo dwóch rodzajów złączy szeregowych: 9- oraz 25-końcówkowych typu DB-9 i odpowiednio DB-25. W komputerach zaopatrzonych w nowsze płyty główne spotyka się jedynie złącza DB-9. Podobnie, wersji DB-25 nie spotyka się już w nowoczesnych urządzeniach pomiarowych. Na rysunku 2.3 przedstawiono wygląd obydwu rodzajów tych złączy.



Rysunek 2.3. *Końcówki DB-9 i DB-25. Ciemne kółka oznaczają nieaktywne wyprowadzenia*

Wykaz sygnałów wykorzystywanych obecnie w interfejsie RS 232C z uwzględnieniem przedstawionych typów złączy podano niżej. Linia 23 DSRD (*Data Signal Rate Detector*) w złączu DB-25 nie została uwzględniona w poniższym zestawieniu, gdyż obecnie praktycznie nie jest wykorzystywana.

Tabela 2.2. *Wykaz sygnałów wykorzystywanych w RS 232C*

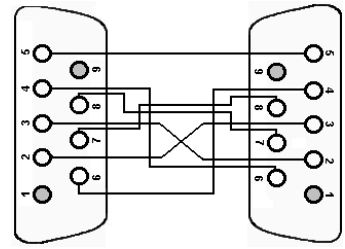
DB-25	DB-9	Opis sygnału	Kierunek sygnału
1	—	PG *Protective Ground* — masa ochronna	—
2	3	TxD *Transmitted Data* — dane wysyłane	Wyjście DTE (PC)
3	2	RxD *Received Data* — dane odbierane	Wejście DTE
4	7	RTS *Request To Send* — żądanie nadawania. PC zgłasza do urządzenia gotowość odbioru danych	Wyjście DTE
5	8	CTS *Clear To Send* — gotowość do wysyłania danych. Urządzenie potwierdza przyjęcie sygnału RTS	Wejście DTE
6	6	DSR *Data Set Ready* — odbiornik gotowy do odbioru danych wysłanych przez komputer	Wejście DTE
7	5	SG *Signal Ground* — masa sygnałowa	—
8	1	(RLSD) DCD *Data Carier Detect* — odbiór fali nośnej. Linia wykorzystywana głównie przez modemy	Wejście DTE
20	4	DTR *Data Terminal Ready* — gotowość komputerado odbierania-wysyłania danych	Wyjście DTE
22	9	RI *Ring Indicator* — wskaźnik wywołania. Linia wykorzystywana głównie przez modemy	Wejście DTE

Linie TxD oraz RxD są przeznaczone do obustronnego przesyłania danych. Nazywamy je liniami danych. Pozostałe zaś, są liniami sterującymi lub kontrolnymi (oczy-wiście za wyjątkiem linii masy). Ogólnie sygnały przekazywane łączem RS 232C można podzielić na trzy grupy:

1. sygnały danych: RxD, TxD.
2. sygnały sterujące urządzeniem zewnętrznym: RTS, DTR.
3. sygnały odbierane od urządzenia (kontrolne): CTS, DSR, RI, RLSD (DCD).

RTS-CTS handshaking

Jak już wspomniano, zamierzonym było, aby książka ta jak najmniej dotyczyła wątków historycznych łącza szeregowego. Pełny opis funkcji linii magistrali interfejsu RS 232C Czytelnik może znaleźć w bogatej literaturze przedmiotu. Nie będziemy się również szczegółowo zajmować poszczególnymi trybami transmisji szeregowej. Dla nas tak naprawdę istotnym będzie tryb półdupleksowy z potwierdzeniem odbioru (transmisja dwukierunkowa naprzemienna), tzw. *handshaking*. W tym trybie komputer i urządzenie mogą naprzemiennie nadawać i odbierać, wykorzystując jeden logiczny kanał danych. Jest to metoda pytanie-odpowiedź. Należy przyznać, że sposób ten jest najprostszym i najskuteczniejszym środkiem wyegzekwowania interesującej nas informacji. Aby zrealizować taką prawdziwą konwersację pomiędzy komputerem a urządzeniem, wystarczy wykorzystać dwie linie danych RxD i TxD oraz dwie linie sterujące RTS i CTS z magistrali RS 232C. Sygnał RTS musi być stale aktywny lub przełączany do tego stanu przed rozpoczęciem nadawania. Podobnie, nadawanie znaków może nastąpić tylko wówczas, gdy sygnał CTS będzie włączony. Taki sposób sprzętowej kontroli transmisji nazywany jest *Hardware flow control* lub *RTS-CTS handshaking* lub jako *Out-of-Band flow control* — sygnały sterujące są generowane i sprawdzane niezależnie od sygnałów danych. Dostępny jest on w większości współczesnych systemów komunikacyjnych. Ideę takiej konwersacji przedstawimy na przykładzie połączenia pomiędzy dwoma złączami DB-9.



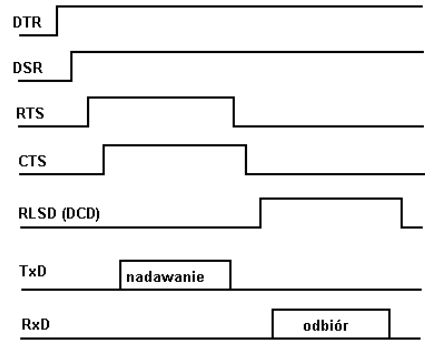
Rysunek 2.4. *Przykład połączenia poprzez złącza DB-9 komputera z urządzeniem pracującym w pełnym trybie półdupleksowym*

Linia wejściowa TxD (3) komputera połączona jest z linią wejściową RxD (2) urządzenia. Linie te służą obustronnej wymianie danych. Za pomocą sygnałów RTS (7)-CTS (8) dokonywany jest wybór aktualnego kierunku transmisji. Po uaktywnieniu łącza szeregowego linia DTR (4) zostaje włączona. W odpowiedzi urządzenie aktywuje linię DSR (6), sygnalizując gotowość do współpracy. Komputer, chcąc przesłać dane do urządzenia, aktywuje swój sygnał RTS (7), czekając na potwierdzenie od urządzenia na linii CTS (8). Jeżeli została ona uaktywniona, komputer wysyła dane linią TxD. Po zakończeniu transmisji linia RTS jest wyłączana (OFF), na co urządzenie odpowiada

również przełączeniem linii CTS do stanu nieaktywnego (OFF). Ogólnie rzecz biorąc, dane przesyłane do komputera linią RxD będą odbierane wówczas, gdy linie DSR i RLSD (DCD) (1) będą włączone. Patrząc na rysunek 2.4 można zauważyć, że jest to typowy układ połączeń stosowany przy współpracy pomiędzy dwoma urządzeniami DTE. Jak to się ma do klasycznego połączenia DTE-DCE? Należy przyznać, że z owym klasycznym sposobem łączenia ma to niewiele wspólnego. Przyczyna jest prosta — nowoczesne urządzenia pomiarowe zaopatrzone są już w programowalne jednostki arytmetyczno-logiczne i tak na dobrą sprawę są już same w sobie komputerami. Innych produktów nie spotyka się obecnie zbyt często na rynku. Jeżeli w instrukcji obsługi kupionego urządzenia nie umieszczono schematu podobnego do tego z rysunku 2.4, oznaczać to będzie, że sprzedawca zrobił kolejny dobry interes, pozbywając się magazynowych zapasów.

Rysunek 2.5 przedstawia pełną sekwencję stanów linii interfejsu RS 232C. Należy jednak zwrócić uwagę, że w większości spotykanych obecnie przypadków DSR i DTR pozostają zwarte, gdyż nowoczesne urządzenia pomiarowe w ogóle nie posługują się linią DSR, zaś linia DCD wykorzystywana jest przeważnie przez modemy.

Rysunek 2.5.
Pełna sekwencja stanów linii interfejsu RS 232C

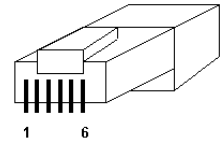


Jeżeli jednak chcielibyśmy z niej zrobić użytek, należy wejście DCD komputera połączyć z linią wyjściową DTR urządzenia. Analogicznie jak na rysunku 2.4 można połączyć ze sobą dwa komputery. W tym przypadku układ połączeń można jeszcze uprościć, zwierając linie RTS-CTS oraz DSR-DTR.

W nowoczesnych przyrządach pomiarowych coraz częściej spotyka się złącze modułowe RJ-11, charakteryzujące się niewielkimi rozmiarami i prostotą montażu przewodów we wtyczkach. Bardzo często wykorzystuje się tu jedynie dwa sygnały dla potrzeb transmisji asynchronicznej.

Na rysunku 2.6 pokazano wygląd takiego złącza, natomiast w tabeli 2.3 specyfikację najczęściej wykorzystywanych przez nie sygnałów.

Rysunek 2.6.
Złącze modułowe RJ-11



Należy jednak zauważyć, że niekiedy występują tu pewne rozbieżności. Niektórzy producenci nieco odmiennie definiują linie sygnałowe w RJ-11 (tabela 2.4), ale zawsze jest to wyraźnie zaznaczone w instrukcji obsługi przyrządu.

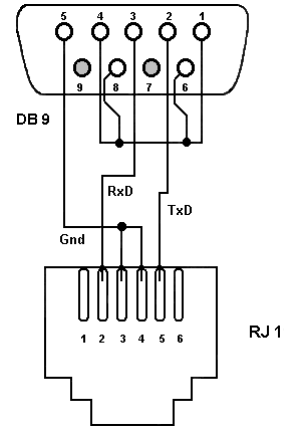
Tabela 2.3. *Specyfikacja linii sygnałowych złącza RJ-11 wg LakeShore*

RJ-11	Opis sygnału
1	RxD *Received Data* — dane odbierane
2	RxD *Received Data* — dane odbierane
3	Gnd *Ground* — masa sygnałowa
4	Gnd *Ground* — masa sygnałowa
5	TxD *Transmitted Data* — dane wysyłane
6	TxD *Transmitted Data* — dane wysyłane

Tabela 2.4. *Specyfikacja linii sygnałowych złącza RJ-11 wg niektórych producentów*

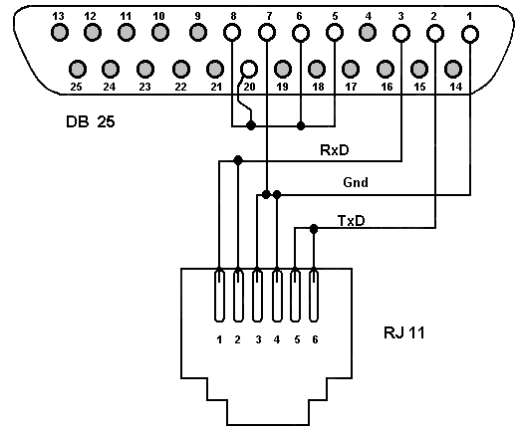
RJ-11	ComputOne	SPECIALIX
1	DTR	RLSD (DCD)
2	TxD	RxD
3	Gnd	DTR / RTS
4	RLSD (DCD)	Gnd
5	RxD	TxD
6	RTS	CTS

Na rysunkach 2.7 i 2.8 pokazano przykłady możliwego układu połączeń pomiędzy komputerem zaopatrzonym w złącze DB-9 lub DB-25 oraz nowoczesnym urządzeniem wyprowadzającym sygnały poprzez RJ-11. Tutaj również widać, że linia wyjściowa TxD komputera połączona jest z odpowiednią linią wejściową RxD urządzenia.



Rysunek 2.7. *Przykład nowoczesnego układu połączeń złączy DB-9 oraz RJ-11 stosowanego przez LakeShore*

Pozostałe linie po stronie komputera pozostają zwarte lub niewykorzystane. Nowoczesne urządzenia nie wykorzystują zbyt wielu linii sterujących. Linie DTR (4), CTS (8), DSR (6) i DCD (1) pozostają zwarte. Linia RTS (7), żądanie nadawania, nie jest wykorzystywana. Ktoś mógłby zapytać, jak będzie wybierany kierunek transmisji. Taki sposób podłączenia, jaki przedstawiono na przykładzie rysunków 2.6 i 2.7 natychmiast sugeruje, że mamy do czynienia z inteligentnym urządzeniem „znaczącym" koniec wysyłanych przez siebie danych parą znaków CR LF, tzw. „*terminatorem*".

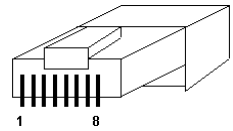


Rysunek 2.8. *Przykład nowoczesnego układu połączeń złączy DB-25 oraz RJ-11 stosowanego przez LakeShore*

Program obsługujący taki przyrząd, napotykając przy odbiorze danych znaki CR LF, będzie już wiedział, że właśnie otrzymał kompletną informację i ewentualnie należy przejść do nadawania.

Jeżeli tylko potrafimy w ten sam lub inny chytry sposób odpowiedzieć miernikowi, otrzymamy bezbłędną obustronną transmisję. Taka metoda programowej kontroli transmisji określana jest mianem *Software flow control.* Nieustanne śledzenie poziomów wielu sygnałów pojawiających się na złączu RS przechodzi powoli do historii. Jedyną trudnością do przezwyciężenia będzie wówczas problem zbudowania odpowiedniej aplikacji. Nieco inną specyfikację sygnałów w złączu RJ-11 podają ComputOne oraz SPECIALIX.

Wielu producentów stosuje również złącza modułowe RJ-45. Poniżej zostały przedstawione niektóre stosowane rozwiązania.



Rysunek 2.9. *Złącze modułowe RJ-45*

Tabela 2.5. *Specyfikacja linii sygnałowych złącza RJ-45 wg niektórych producentów*

RJ-45	Comput-One	Chase	DIGIBOARD	MICRO-ANEX1	MICRO-ANEX2	EQUINOX	IBM
1	—	DCD	DSR	RTS	—	RTS	CTS[2]
2	RTS	RTS	RTS	DTR	DTR	DTR	DTR[2]
3	RxD	PG[1]	PG[1]	TxD	TxD	RxD	TxD[2]
4	DCD	TxD	TxD	DCD	—	Gnd	Gnd[2]
5	Gnd	RxD	RxD	RxD	RxD	TxD	DCD[2]
6	TxD	Gnd	Gnd	Gnd	Gnd	Gnd	RxD[2]
7	DTR	CTS	CTS	DSR	—	DSR	RTS[2]
8	CTS	DTR	DTR	CTS	CTS	CTS	—

[1] Linia PG zwykle połączona jest z obudową. Jako potencjał odniesienia używana jest linia Gnd.

[2] Identycznie jak w ComputOne, odwrócono tylko numerację styków.

Konwertery interfejsu RS 232C

Istnieje wiele rodzajów konwerterów sygnałów interfejsu RS 232C na inne standardy RS. Do najczęściej stosowanych należą układy służące do łączenia urządzeń wyposażonych w interfejs RS 232C z urządzeniami wyposażonymi w interfejs RS 485 lub RS 422. Transmisja szeregowa w standardach 485 lub 422 jest dużo szybsza i bardziej odporna na zakłócenia, zapewnia ponadto większy zasięg transmitowanych sygnałów.

W standardzie RS 485 szeregowa transmisja danych cyfrowych odbywa się przez dwuprzewodową symetryczną linię transmisyjną, do której można dołączyć nawet 32 nadajniki i odbiorniki. Stosując odpowiednie powielacze sygnału, liczbę takich urządzeń można znacznie zwiększyć. Interfejs ten umożliwia realizację wielopunktowej transmisji w trybie półdupleksowym. Standard elektryczny RS 422 nie różni się w istocie od RS 485. Różnica polega na możliwości dołączenia do jednej pary przewodów jednego nadajnika nawet do 10 odbiorników. W układzie RS 422 możliwa jest transmisja w trybie pełnego dupleksu, czyli w modzie jednoczesnego nadawania i odbioru danych.

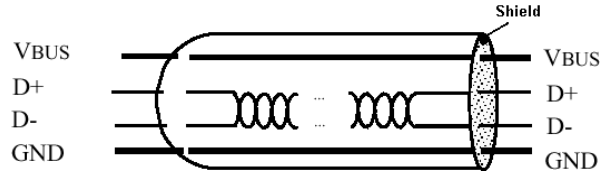
Konwertery interfejsu RS 232C konstruowane są w postaci niewielkich pudełek zawierających z jednej strony złącze DB-25 lub DB-9 do podłączenia do łącza RS 232C w komputerze, zaś z drugiej strony inne złącze, np. PHOENIX, do podłączenia linii i napięcia zasilającego. Układy te, zapewniając izolację galwaniczną łączonych urządzeń i linii transmisyjnej, z reguły zasilane są oddzielnym zasilaczem stabilizowanym. Stosując tego rodzaju konwertery sygnałów, możliwe jest uzyskanie szybkości transmisji w granicach 2,5 Mb/s przy maksymalnej długości linii około 1200 m, co wydaje się rozwiązaniem w pełni zadawalającym nawet w warunkach przemysłowych. Zestaw dwóch konwerterów może być z powodzeniem stosowany do realizacji połączenia pomiędzy dwoma urządzeniami zaopatrzonymi w interfejs RS 232C.

Konwertery USB/RS 232C

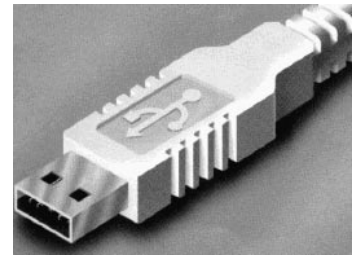
Konwertery tego typu pozwalają na podłączenie urządzeń zaopatrzonych w interfejs szeregowy RS 232C do portów USB w komputerze. Wśród dostępnych na rynku wysoką kulturą działania oraz niezawodnością wyróżniają się urządzenia gliwickiej firmy YUKO. Konwertery takie posiadają złącze USB typu Am (A-mini) oraz DSUB 25 (port RS 323C), w którego obudowie znajduje się układ konwertera zasilanego z portu USB komputera. Złącza te połączone są odcinkiem kabla długości ok. 90 cm. Dostarczane wraz z konwerterem sterowniki instalują się w 32 bitowych systemach Windowsowych (98, ME, NT, 2000 oraz XP), dzięki czemu uzyskujemy dostęp do dodatkowego 8-bitowego portu o dowolnie zadeklarowanej wartości od COM1 do COM256, którego można używać tak jak portu standardowego. Należy zwrócić uwagę, iż jest to jednak port wirtualny, dlatego programy, które bezpośrednio obsługują porty komunikacyjne (np. MS DOS) nie będą działać poprawnie. W przeciwieństwie do standardowych portów COM, port konwertera jest bardzo szybki. Dostarczane sterowniki zapewniają transmisję danych do 921 kb/s z możliwością rozszerzenia w przypadku transmisji asynchronicznej nawet do wartości 2 Mb/s.

Na rysunkach 2.10, 2.11 oraz 2.12 pokazano odpowiednio schemat przekroju czterożyłowego kabla wykorzystywanego przez USB, wygląd złącza typu A wykorzystywanego do podłączenia urządzeń z komputerem (hostem, ang. *upstream port*) oraz numerację sygnałów USB, których wykaz zawiera tabela 2.6.

Rysunek 2.10. *Czterożyłowy kabel USB*



Rysunek 2.11. *Złącze USB typu A*



Rysunek 2.12. *Numeracja styków złącza USB typu A*

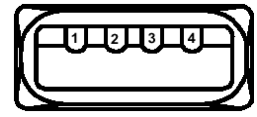


Tabela 2.6. *Wykaz sygnałów wykorzystywanych w USB*

Numer	Oznaczenie	Opis sygnału	Kolor przewodu
1	V_{BUS}	Przewód zasilania (max. +5,25 [V])	Czerwony
2	Data–(D–)	Przewód symetrycznej skrętki sygnałowej	Biały
3	Data+(D+)	Przewód symetrycznej skrętki sygnałowej	Zielony
4	GND	Przewód masy zasilania	Czarny
Shell	Shield	Ekran	

Protokół XON-XOFF

Wiele urządzeń wymaga stosowania programowej kontroli przepływu danych z wykorzystaniem tzw. protokołu XON-XOFF. Przykładem praktycznego wykorzystania niektórych znaków pokazanych w tabeli 2.1 jest właśnie ten protokół, czasami nazywany DC1-DC3 lub ^Q-^S.

Jeżeli dane przychodzą zbyt szybko do odbiornika i urządzenie odbierające nie może ich tak szybko pobierać z bufora wejściowego, program sterujący może wysłać znak XOFF (DC3 lub dziesiętnie 19 albo Control + S). Urządzenie nadające zatrzymuje dalszą transmisję (jeżeli oczywiście „wie”, co to jest XOFF), dopóki od strony odbiornika nie nadejdzie znak XON (DC1 lub dziesiętnie 17 albo Control + Q).

Jeżeli jednak XOFF zostanie wysłany zbyt późno, może nastąpić przepełnienie bufora wejściowego. Podobnie opóźnienie wysłania XON z reguły powoduje zablokowanie portu komunikacyjnego.

We współczesnych, inteligentnych urządzeniach o wysokim stopniu wzajemnej synchronizacji protokół ten przewiduje wysłanie XOFF, jeżeli bufor wejściowy jest wypełniony powyżej ¾ deklarowanego rozmiaru. Program sterujący urządzeniem wysyła znak XON, jeżeli bufor jest wypełniony mniej niż w ½. W tym przypadku transmisja musi przebiegać w pełnym trybie dupleksowym. Dane przekazywane są jednocześnie w obu kierunkach niezależnie od siebie, po oddzielnych liniach transmisyjnych (patrz tabela 2.3). Wykorzystujemy tu dwie pary linii RxD i TxD (RTS i CTS nie mają znaczenia), linia DTR może (ale nie musi) być wykorzystywana do włączania i wyłączania urządzenia. Windows podtrzymuje ten protokół, o czym Czytelnik może się przekonać, zaglądając do *Panelu sterowania* i *Właściwości portów komunikacyjnych* oraz czytając dalej tę książkę.

Protokół ENQ-ACK

Jest to obecnie jedna z rzadziej stosowanych metod kontroli przepływu danych w urządzeniach pomiarowych. Urządzenie transmitujące wysyła regularnie zapytanie ENQ (patrz tabela 2.1) po każdej, wcześniej ustalonej porcji transmitowanych danych. Kiedy „odbierający" jest gotowy do przyjęcia kolejnego bloku informacji, wysyła do „nadającego" potwierdzenie ACK, sygnalizując tym samym gotowość przyjęcia następnej porcji danych. W metodzie tej z reguły nie kontroluje się stopnia wypełnienia bufora pamięci. Jeżeli „nadający" po wysłaniu kolejnej porcji informacji nie otrzyma potwierdzenia jej odbioru, zaczyna wysyłać przez pewien, ściśle ustalony czas, znak, np. LF, dając odbiornikowi możliwość ewentualnego przetworzenia wcześniej otrzymanych danych. Jeżeli po określonym czasie nie nadejdzie potwierdzenie ACK, nadajnik wstrzymuje dalszą transmisję do czasu jego otrzymania.

Protokół ETX-ACK

Protokół ten funkcjonuje nieco inaczej. Komputer wysyła cały blok danych zakończonych znakiem ETX. Odbiornik wysyła do komputera znak potwierdzenia ACK dopiero po całkowitym przetworzeniu odebranej porcji danych. Zastosowanie tego protokołu wymaga istnienia w odbiorniku wystarczająco dużego bufora pamięci, aby transmisja nie była zatrzymywana przed odebraniem całej porcji informacji.

Protokół SOH-ETX

Protokół SOH-ETX bywa coraz częściej wykorzystywany w tzw. urządzeniach wielozakresowych, gdzie zwracana informacja obejmuje nie tylko np. aktualną wartość mierzonego prądu przemiennego, ale jeszcze inne jego parametry. Początek nagłówka

tekstu lub aktualny adres odpytywanego urządzenia jednoznacznie określa znak SOH (ale nie STX, gdyż tekst taki może być poprzedzony specjalnymi znakami formatowania lub znakami nie będącymi tekstem). Koniec przesyłanej porcji danych wyraźnie określa znak ETX (*End of Text* por. tabela 2.1).

Protokoły typu master-slave

Wykorzystując protokoły typu master-slave tylko jednostka nadrzędna (ang. *master*) może inicjować transmisję danych w systemie, zaś jednostki podrzędne (ang. *slave*) jedynie odpowiadają na zdalne zapytania. Ciąg wykonywanych czynności, które pod groźbą utraty spójności danych muszą być wykonane w całości, tzw. transakcja, składa się z polecenia wysyłanego z jednostki master do slave oraz z odpowiedzi przesyłanej z jednostki slave do master. Odpowiedź zawiera dane żądane przez jednostkę nadrzędną lub potwierdzenie realizacji odebranego polecenia. Master może indywidualnie odpytywać szereg jednostek podrzędnych lub też przesyłać wiadomości rozgłoszeniowe (ang. *broadcast*), przeznaczone dla wszystkich urządzeń podrzędnych w systemie. Na polecenia rozgłoszeniowe jednostki podrzędne z reguły nie przesyłają odpowiedzi. Protokoły tego rodzaju wykorzystywane są obecnie m.in. przez interfejsy typu MODBUS, które wykorzystują asynchroniczną transmisję znakową zgodną ze standardem RS 232C.

Rola oprogramowania, a podstawowe funkcje interfejsu

Aby zrealizować prawidłową wymianę informacji pomiędzy komputerem a urządzeniami zewnętrznymi, dla których pełnić on będzie rolę kontrolera, należy w pierwszej kolejności określić funkcje szeroko rozumianego interfejsu w tym systemie. W dobrze zaprojektowanym układzie interfejs powinien spełniać następujące wymagania:

1. Zapewnienie właściwego sposobu inicjalizacji połączenia.

   Aby uzyskać dostęp do urządzenia przyłączonego do portu komunikacyjnego, port ten należy fizycznie uaktywnić — otworzyć do transmisji. W standardowym PC mamy z reguły do dyspozycji tylko dwa szeregowe porty komunikacyjne. Często zachodzi jednak konieczność obsłużenia wielu urządzeń. Stosuje się wówczas specjalne karty lub konwertery, umożliwiające uzyskanie dostępu do większej ich liczby. Przyrządy podłączone do portów komunikacyjnych muszą być rozróżnialne, jeżeli chcemy nimi naprawdę sterować. Będą takimi, jeżeli aplikacja będzie w stanie rozróżnić poszczególne porty szeregowe i kierować do nich odpowiednie komunikaty oraz odbierać dane. Właściwa inicjalizacja portu polega na nadaniu mu unikalnego identyfikatora, którym można swobodnie operować w trakcie działania programu.

2. Zapewnienie właściwej synchronizacji transmitowanych danych pomiędzy komputerem i urządzeniami zewnętrznymi oraz udostępnienie metod natychmiastowej i automatycznej korekty różnego rodzaju błędów, pojawiających się w czasie transmisji.

Większość standardowych interfejsów ma wbudowane funkcje synchronizacji, które w pewnym stopniu mogą minimalizować pojawiające się opóźnienia w kanale transmisyjnym — jedną z głównych przyczyn powstawania błędów. Rolą oprogramowania będzie ich umiejętne wyzwalanie. Aplikacja zarządzająca transmisją musi być skonstruowana w sposób zapewniający bezbłędne funkcjonowanie systemu pomiarowego lub komunikacyjnego. Właściwa reakcja na pojawiające się w czasie transmisji błędy oraz możliwość ich ewentualnej korekcji są zawsze istotnymi elementami programu komunikacyjnego.

3. Zapewnienie właściwej kontroli transmisji oraz wyboru jej kierunku.

   Kontrolowanie aktualnego kierunku transmisji może być realizowane sprzętowo lub programowo. Na pewno bardziej przydaną jest umiejętność programowej kontroli przepływu danych. Użytkownik danego systemu wie najlepiej, jakie dane i w jakim czasie chce otrzymać od urządzenia. Aplikacja obsługująca dany interfejs musi być tak zaprojektowana, aby możliwym był „płynny" wybór kierunku nadawanie-odbiór. W tym miejscu należy zwrócić szczególną uwagę na to, by nie tracić danych w momencie zmieniania kierunku transmisji. Stosując metodę buforowania danych, program musi być wyczulony na możliwość odbierania swoich własnych komunikatów przy nagłej zamianie ról z nadajnika na odbiornik.

4. Udostępnienie możliwości zatrzymania transmisji w dowolnym momencie bez ryzyka utraty danych.

   Tę właściwość same interfejsy posiadają tylko w ograniczonym stopniu. Program kontrolujący transmisję jest naprawdę funkcjonalny wówczas, gdy zawiera opcje umożliwiające czasowe wstrzymanie operacji odbioru-nadawania bez ryzyka utraty informacji. Jest to szczególnie ważne w przypadku aplikacji wielowątkowych lub generujących własne przerwania systemowe. Właściwość tę musi uwzględniać oprogramowanie sterujące jednocześnie portem szeregowym oraz różnego rodzaju kartami przetwornikowymi zaopatrzonymi w przetworniki analogowo-cyfrowe. W obecnych komputerach procesor programuje zaledwie kilka rejestrów sterujących urządzenia, wysyłając rozkaz wykonania pewnej operacji (np. odebranie znaku przez port szeregowy). Istnieją przynajmniej dwa sposoby poinformowania procesora o tym, że dana operacja właśnie się zakończyła. Po pierwsze można zastosować tzw. *polling*, gdzie procesor wysyła regularne zapytania do urządzenia. Częstotliwość tych zapytań jest kontrolowana przez aktualnie działającą aplikację. Jednak w praktyce bardzo trudno jest określić optymalną częstotliwość takiego odpytywania i z tego względu sposób ten jest bardzo niewygodny. Drugi sposób polega na tym, że samo urządzenie zgłasza wykonanie danego zadania. W stosownym momencie procesor przerywa wykonywanie aktualnego programu, pamiętając stan swoich rejestrów uaktywnia funkcje reagujące na zgłoszenie danego urządzenia. Mówimy wówczas, że nastąpiło przerwanie sprzętowe *interrupt* pochodzące od urządzenia. Ten sposób sterowania przepływem danych w PC jest stosowany wszędzie tam, gdzie mamy do czynienia z intensywnym i nieregularnym przepływem danych pomiędzy urządzeniami, a pamięcią operacyjną. W praktyce mamy możliwości programowej kontroli aktualnie występujących przerwań. Jednak w rzeczywistości jest to zadanie uciążliwe i lepiej jest wykorzystać zalety programowania obiektowo-zdarzeniowego.

5. Zapewnienie możliwości odpowiedniego odbierania, przechowywania i wysyłania danych.

   Standardowe interfejsy mają możliwość buforowania danych. Jest to zaleta, którą doceniamy wtedy, gdy nie jesteśmy w stanie w sposób ciągły odbierać przychodzących danych lub nie możemy ich wysłać w ściśle określonym momencie. Oprogramowanie sterujące przepływem danych pełni w takich przypadkach rolę wspomagającą. Aplikacja powinna umieć odczytać aktualny stan bufora wejściowego i zdecydować o pobraniu z niego interesujących nas danych. Odbierając informacje w sposób ciągły, należy nieustannie kontrolować bufor danych, nawet w sensie fizycznej ingerencji. Dobrze działający program nie może dopuścić do jego przepełnienia, gdyż grozi to całkowitą utratą informacji. Rolą oprogramowania będzie również odpowiednie czyszczenie bufora danych w trakcie transmisji. Jest to zawsze punkt newralgiczny systemu komunikacyjnego.

W celu ujednolicenia i uproszczenia sposobów projektowania oprogramowania wykorzystywanego w różnych systemach pomiarowych stworzono standard opisujący zestaw uniwersalnych instrukcji programujących urządzenia pomiarowe, tzw. język SCPI (ang. *Standard Commands for Programmable Instruments*). Zdefiniowane są tam wszystkie ujednolicone przez producentów urządzeń pomiarowych instrukcje (rozkazy) umożliwiające zaprogramowanie nowoczesnego przyrządu w zależności od wykonywanego przezeń zadania. Programiści dostali więc do dyspozycji uniwersalny język zapytań i odpowiedzi, należy tylko umiejętnie go wykorzystać. Niestety, to niewątpliwe udogodnienie nie zwalnia nas z konieczności samodzielnego stworzenia (lub kupna) aplikacji, potrafiącej wykorzystać zalety SCPI. Czytelników pragnących poszerzyć swoje wiadomości na ten temat odsyłam do książki Wojciecha Mielczarka *Urządzenia pomiarowe i systemy kompatybilne ze standardem SCPI*, wyd. Helion (1999).

Podsumowanie

W rozdziale tym zostały zaprezentowane podstawowe wiadomości dotyczące szeregowej transmisji asynchronicznej oraz standardu RS 232C. Tematy te zostały potraktowane w sposób zwięzły, ale zupełnie wystarczający do zrozumienia zagadnień związanych z programową kontrolą łączy szeregowych. Przedstawione też zostały rzadko spotykane przykłady nowoczesnych sposobów połączeń coraz częściej stosowane w szeregowej transmisji danych. Omówiono również najważniejsze stosowane obecnie protokoły kontroli transmisji danych. Wymieniono też, jakimi cechami powinny charakteryzować się aplikacje, obsługujące transmisję szeregową.

W książce tej, wraz z wprowadzaniem konkretnych algorytmów mogących obsługiwać komunikację szeregową, omówione zagadnienia będą stopniowo uzupełniane. Bardziej szczegółowe informacje dotyczące standardu RS 323C Czytelnik może znaleźć w bogatej literaturze przedmiotu oraz na licznych stronach *WWW*. Osobom pragnącym poszerzyć swoje wiadomości na temat szeregowych interfejsów cyfrowych polecamy książkę Wojciecha Mielczarka *Szeregowe interfejsy cyfrowe*, wyd. Helion (1994).