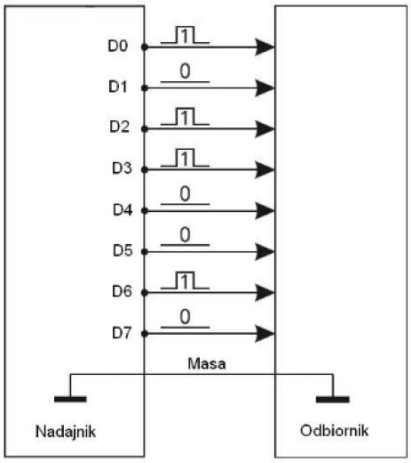
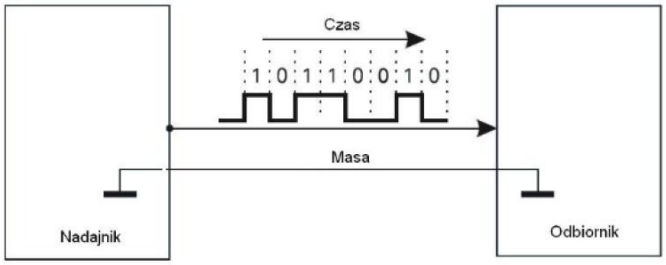
TMiSW6 Interfejsy szeregowe

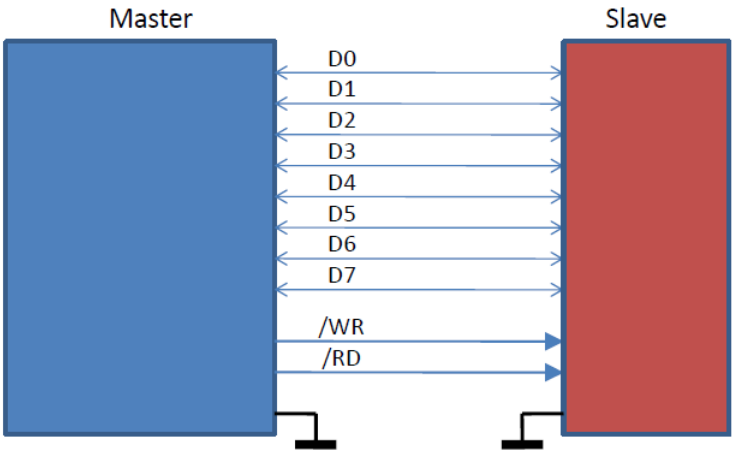
Interfejs - jest to zespół środków zapewniających dopasowanie mechaniczne, elektryczne i informacyjne, oraz ustalających funkcjonalne relacje pomiędzy fizycznie odrębnymi częściami systemu, zgromadzonym w celu wymiany informacji między nimi.
Zadania i funkcje interfejsu:

- konwersja
- synchronizacja
- przerwania
- buforowania
- zarządzanie
- korekcja błędów

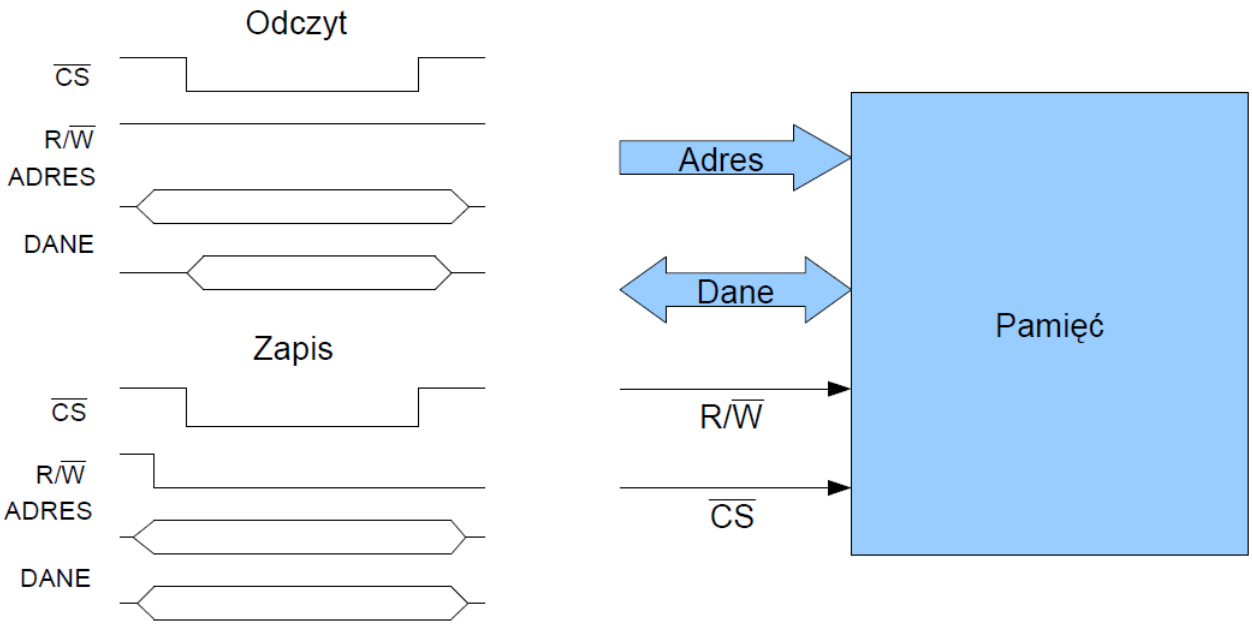




Porównanie transmisji równoległej i szeregowej

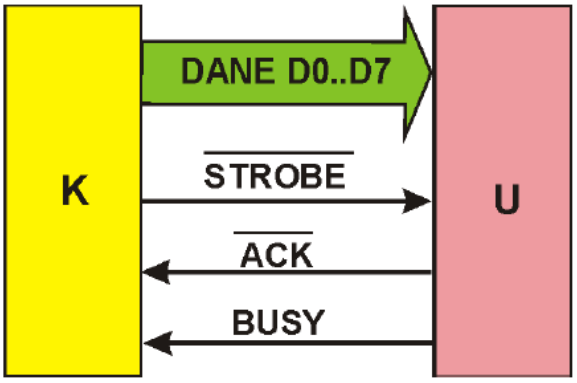


Przykład transmisji równoległej 8-bitowej, dwukierunkowej, zapis i odczyt danych z urządzenia podrzędnego



Interfejs pamięci

Transmisja asynchroniczna z synchronizacją sprzętową



Zalety i wady transmisji szeregowej

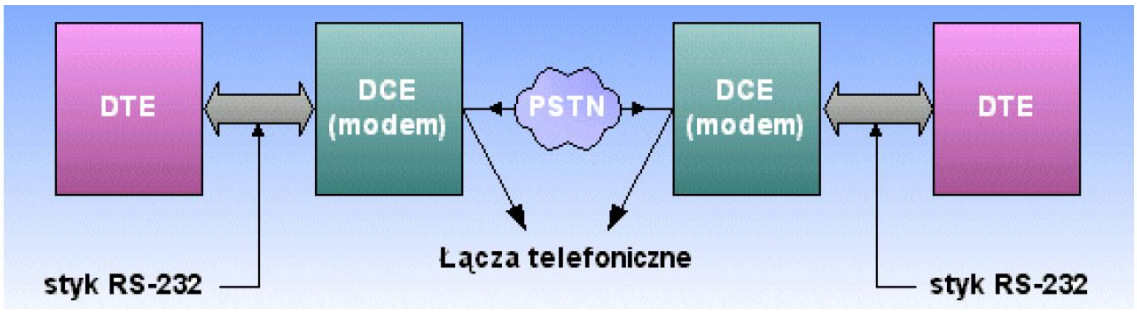
Zalety:

- mała liczba przewodów,
- mniejsze wtyczki, złącza, gniazdka, itp.,
- możliwość transmisji wielobitowej,
- wiele odbiorników/nadajników podłączonych do tych samych przewodów
- większa odporność na zakłócenia, (transmisja różnicowa),
- możliwa transmisja na większe odległości,
- możliwa transmisja bezprzewodowa

Wady:

- ograniczona prędkość transmisji,
- duża częstotliwość taktowania dla osiągnięcia odpowiedniej szybkości transmisji,
- wymagany układ do zamiany na postać równoległą i szeregową, (serializacja danych)
- dodatkowe dane do korekcji błędów,

Standard RS-232 został opracowany na początku lat sześćdziesiątych w celu normalizacji interfejsu pomiędzy urządzeniami wymieniającymi dane. Szczególnie nacisk położono na zdefiniowanie interfejsu pomiędzy terminalem (DTE - Data Terminal Equipment) a modemem (DCE - Data Communication Equipment). Standard ten jest obecnie bardzo często stosowany przy szeregowej transmisji danych pomiędzy różnymi typami urządzeń DTE. Obecnie najbardziej rozpowszechnioną wersją jest wersja oznaczona symbolem RS-232C, która jest powszechnie używana do transmisji danych na nieduże odległości (do 15 m). Interfejs RS-232C występuje w dwu wersjach: 9 linii – wtyk DB-9, 25 linii wtyk DB-25.



Wykorzystanie łącza RS232

Poniżej podano zestawienie sygnałów interfejsu RS-232C używanych w komputerach PC:

Wtyk DB-25	*Wtyk DB-9*	*Sygnał*	*Kierunek transmisji*
1	-	-	-
2	3	TxD	Terminal -> Modem
3	2	RxD	Modem -> Terminal
4	7	RTS	Terminal -> Modem
5	8	CTS	Modem -> Terminal
6	6	DSR	Modem -> Terminal
7	5	-	Mass
8	1	DCD	Modem -> Terminal
20	4	DTR	Terminal -> Modem
22	9	RI	Modem -> Terminal
23	-	DSRD	Modem <-> Terminal

1) **TxD** - Transmitted Data - dane nadawane

2) **RxD** - Received Data - dane odbierane

3) **RTS** - Request To Send - urządzenie sygnalizuje tą linią zamiar przekazywania danych

4) **CTS** - Clear To Send - linią tą przesyłane jest potwierdzenie przyjęcia sygnału RTS i stwierdzenie gotowości do odbioru danych. Para sygnałów sterujących RTS/CTS może przy półdupleksowym trybie pracy łącza sterować kierunkiem transmisji

5) **DSR** - Data Set Ready - specyfikacja RS-232C określa ten sygnał jako meldunek urządzenia (modemu), że zostało nawiązane połączenie i układ jest gotów do przyjęcia danych (w praktyce nie używane i oznacza tylko włączenie urządzenia, modemu)

6) **DTR** - Data Terminal Ready - sygnał ten wskazuje w ogólności na gotowość urządzenia nadającego. Musi on pozostawać aktywny przez cały czas trwania połączenia. Para sygnałów DTR i DSR odpowiada za utrzymanie połączenia

7) **DCD** - Data Carrier Detect - modem sygnalizuje tą linią odbiór fali nośnej, co oznacza, że druga strona jest w trakcie nawiązywania połączenia. Sygnał DCD pozostaje aktywny przez cały czas trwania transmisji

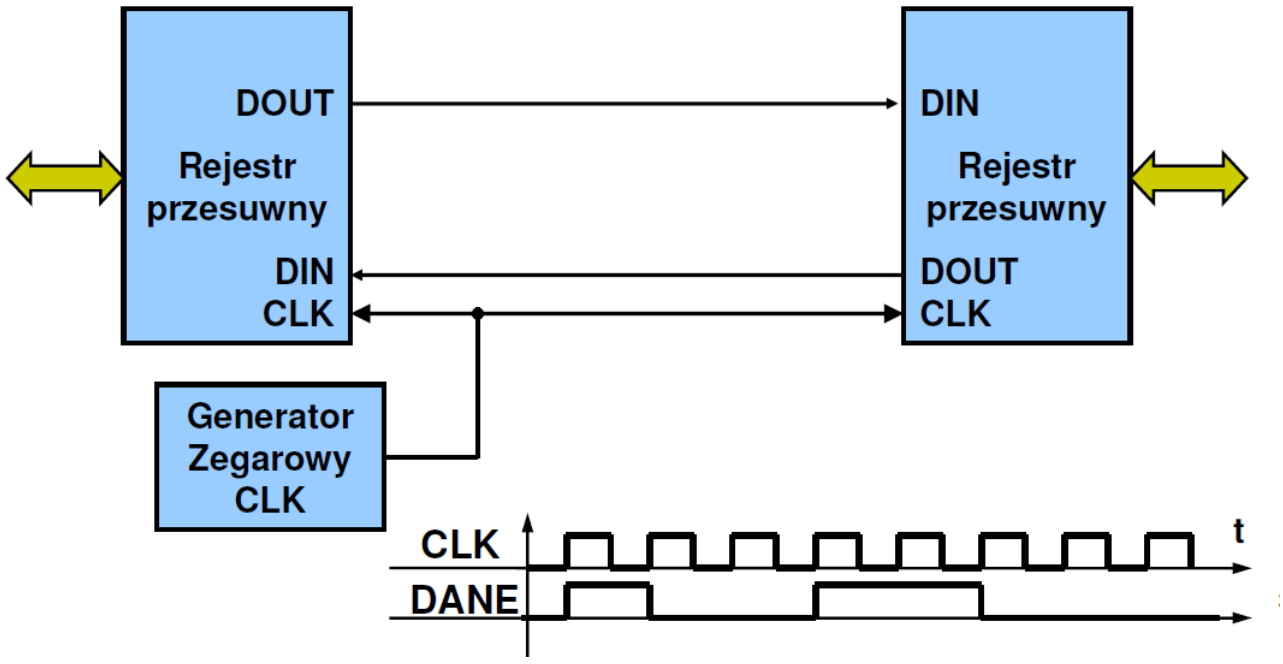
8) **RI** - Ring Indicator - w przypadku połączenia modemów przez sieć telefoniczną urządzenie nadające informowane jest o odebraniu sygnału odpowiadającego wywołaniu abonenta (dzwonieniu)

9) **DSRD** - Data Signal Rate Detector - linia ta występuje tylko w 25-końcówkowej wersji łącza. Umożliwia dostosowanie się korespondentów do jednej z dwóch możliwych prędkości transmisji. Z sygnału tego mogą korzystać obie strony.

Linie TxD i RxD są właściwymi liniami służącymi wymianie danych, pozostałe przewody są wykorzystywane do sterowania transmisją.

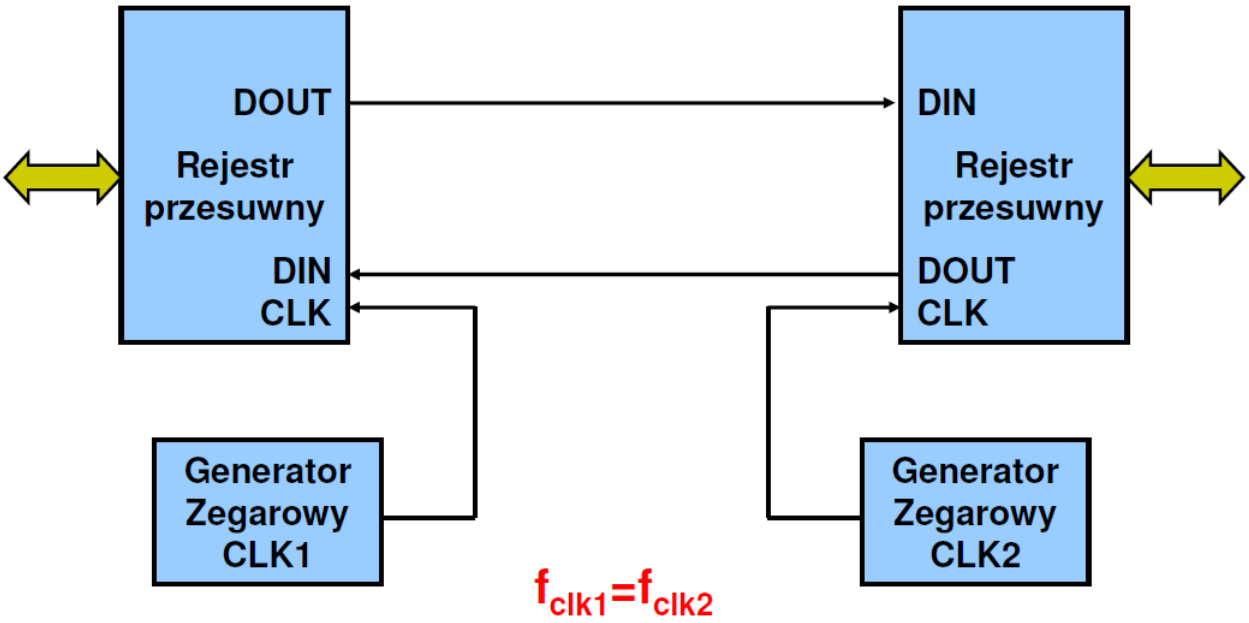
W standardzie RS-232C transmisja danych odbywa się szeregowo bit po bicie, przy czym definiuje się dwa rodzaje transmisji: synchroniczna i asynchroniczna (znakowa).

Idea transmisji szeregowej synchronicznej

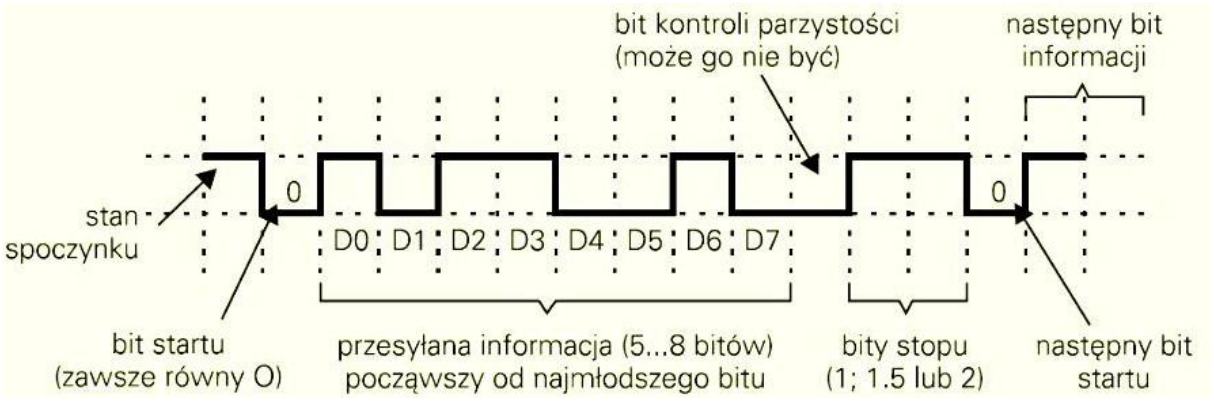


Podczas transmisji synchronicznej, dzięki określonemu impulsowi taktującemu, utrzymywane jest stałe tempo przekazywania informacji. Nie występują tutaj przerwy spowodowane koniecznością synchronizacji pojedynczych porcji informacji, a więc uzyskuje się lepsze wykorzystanie linii łączących. Dane mają strukturę określającą ich przeznaczenie (np. dane do wyświetlenia, dane do drukowania, sterowanie terminalem itp. Zwykle dla kontroli poprawności transmisji pakiet zawiera dodatkowe dane pozwalające zweryfikować poprawność transmisji. W transmisji synchronicznej ilość bitów pomiędzy pakietami nie musi być wielokrotnością bajtu.

Idea transmisji szeregowej asynchronicznej



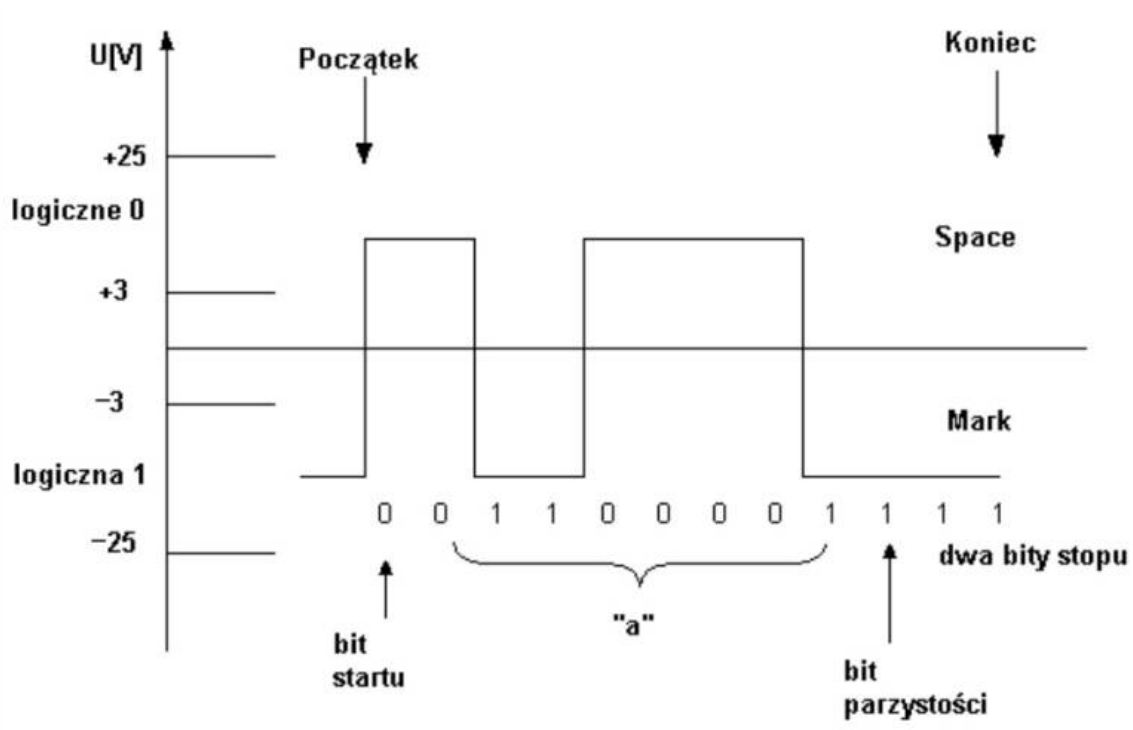
Asynchroniczna transmisja znakowa polega na przesyłaniu pojedynczych znaków, które posiadają ściśle określony format. Początek znaku stanowi bit startu, jałowy z punktu widzenia przesyłanej informacji i służący jedynie celom synchronizacyjnym. Dalej następuje pole danych, na które wprowadza się kolejne bity stanowiące treść znaku. Bezpośrednio za polem danych przewidziano bit kontrolny, służący zabezpieczeniu informacji znajdującej się na polu danych. Transmitowany znak kończy jeden lub dwa bity stopu. Powyższa definicja pozwala na przesłanie jednego znaku. W ramach znaku bity przesyłane są synchronicznie, to znaczy zgodnie z taktem nadajnika. Natomiast kolejne znaki są przesyłane asynchronicznie - ich wyprowadzanie nie jest synchronizowane żadnym sygnałem, a więc odstęp pomiędzy nimi jest dowolny.



Struktura przesyłu pojedynczego znaku (asynchronicznie)

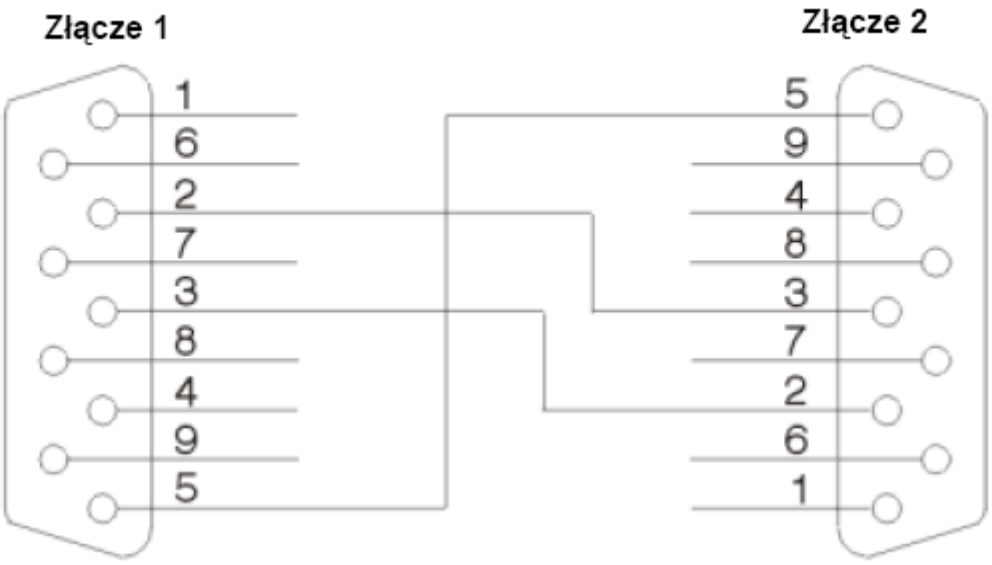
stan spoczynku, bit startu zawsze równy 0, przesyłana informacja (5...8 bitów) począwszy od najmłodszego bitu, bit kontroli parzystości (może go nie być), bit stopu (1; 1.5 lub 2), następny bit startu, następny bit informacji

Standard elektryczny interfejsu RS232C

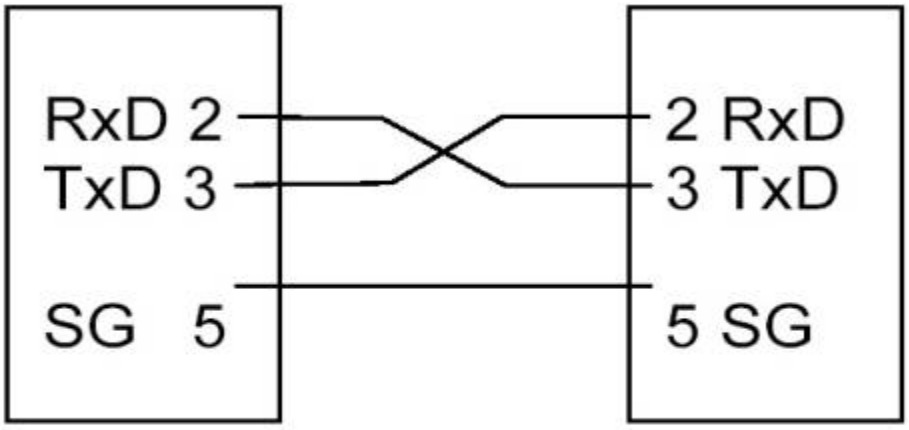


Specyfikacja napięcia definiuje "1" logiczną jako napięcie -3V do -15V, zaś "0" to napięcie +3V do +15V. Poziom napięcia wyjściowego natomiast może przyjmować wartości -12V, -10V, +10V, +12V, zaś napięcie na dowolnym styku nie może być większe niż +25V i mniejsze niż -25V. Aby połączyć dwa komputery przy użyciu łącza szeregowego RS-232 wykorzystuje się tzw. kabel zerowy (null modem cable). Można wyróżnić aż cztery typy takiego połączenia: proste połączenie bez potwierdzenia, połączenie z potwierdzeniem zwrotnym, połączenie z potwierdzeniem częściowym oraz połączenie z pełnym potwierdzeniem. Najczęściej używane jest połączenie poprzez tzw. „kabel zerowy”

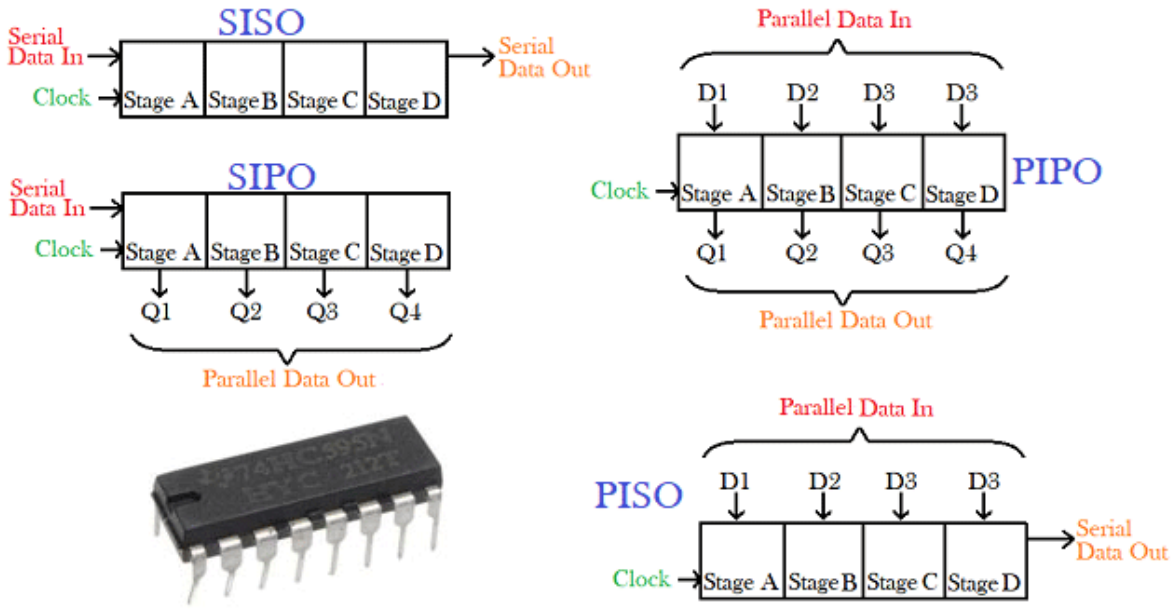
Połączenie modem zerowy



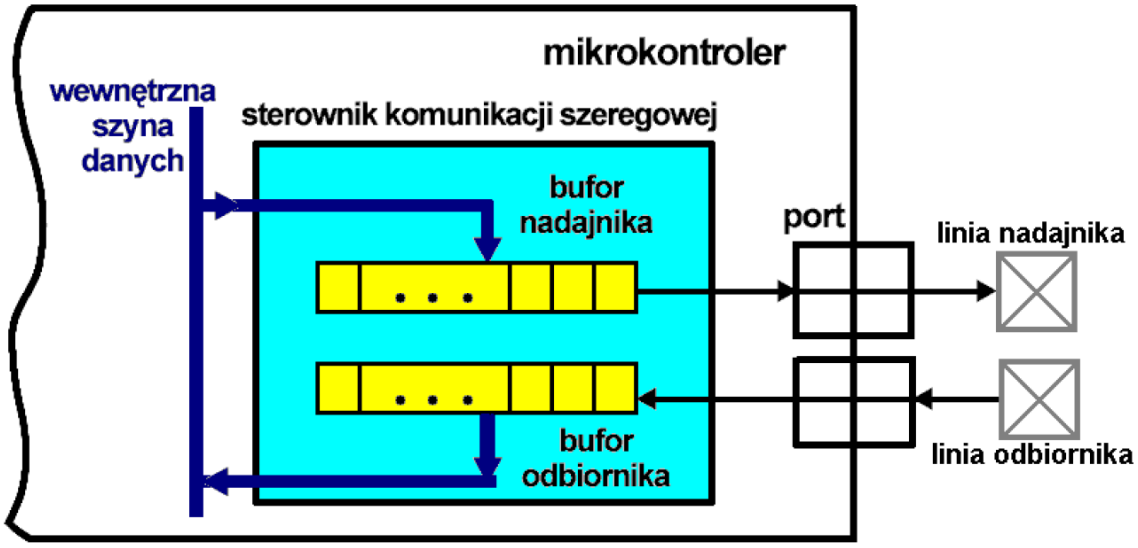
Złącze 1	Złącze 2	Funkcja
2	3	Rx ← Tx
3	2	Tx → Rx
5	5	Signal ground



DTE – DTE *(null modem)*



Sterowniki komunikacji szeregowej służą do wymiany informacji między mikrokontrolerem, a jego otoczeniem. Przesyłanie danych odbywa się w sposób szeregowy.



Urządzenie to umożliwia wysyłanie zawartości określonego rejestru, tzw. **bufora nadajnika**, w postaci szeregowej poprzez określone wyprowadzenie portu. Oznacza to, że na wyjściu linii portu pojawia się ciąg binarny odpowiadający zawartości wysyłanego rejestru (funkcja nadajnika - **transmiter**). W funkcji odbiornika (**reciver**) sterownik komunikacji szeregowej potrafi przetworzyć ciąg binarny doprowadzony do wejścia określonej linii portu na zawartość rejestru, zwanego **buforem odbiornika**. Bufory nadajnika i odbiornika są dostępne z poziomu programu użytkownika.

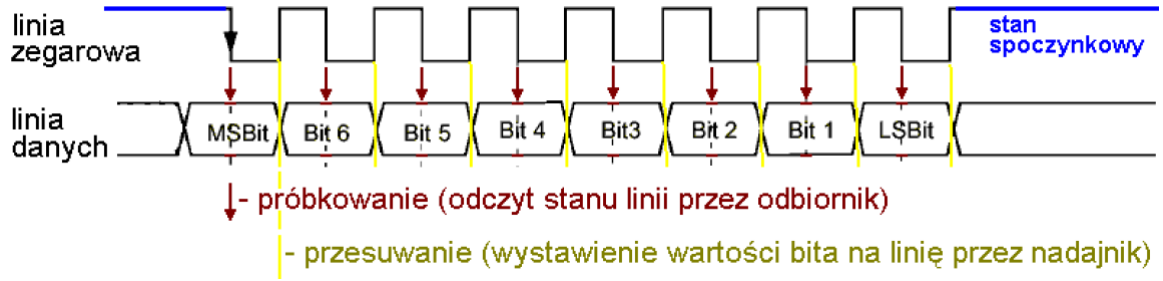
Wyróżnia się dwa rodzaje transmisji szeregowej:

- **asynchroniczną,**
- **synchroniczną.**

Dane przesyłane asynchronicznie nie są związane z żadnym sygnałem synchronizującym, w szczególności nie towarzyszy im sygnał zegara. Transmisja przebiega zwykle bajtami przesyłanymi w postaci ciągu bitów. Oprócz ośmiu bitów danych przesyła się dodatkowo bit startu oraz opcjonalnie bit parzystości do detekcji błędów i bit stopu. Pomiędzy nadajnikiem a odbiornikiem musi być ustalona częstotliwość przesyłania danych. Odbiornik z ustaloną częstotliwością próbkuje swoją linię wejściową danych. Po wykryciu bitu startu (najczęściej stan niski) odczytuje ustaloną liczbę bitów. Transmisja kończy się bitem stopu ustawiającym linię danych w stan nieaktywny.

Przebiegi czasowe transmisji asynchronicznej na linii danych

Przy transmisji synchronicznej równolegle z ciągiem bitów danych przesyła się sygnał synchronizujący (zegarowy), który określa chwile, w których stan linii danych odpowiada ważnym wartościom kolejnych bitów. Po każdym bajcie może być dodatkowo przesyłany bit parzystości.



Przebiegi czasowe transmisji synchronicznej na linii danych i sygnału synchronizującego

Simpleks, half- i full-dupleks

W przypadku transmisji szeregowej istnieją pojęcia:

- simpleks
- half-dupleks (półdupleks)
- full-dupleks (pełny dupleks).

Transmisja w trybie Simpleks występuje wówczas, gdy dane wędrują tylko w jednym kierunku, często bez możliwości odpowiedzi. Typowym przykładem tego typu transmisji jest przekaz sygnału radiowego lub telewizyjnego. Nadajnik nadaje sygnał np. cyfrowy telewizji DVBT, który jest odbierany przed odbiornik, dekodowany i wyświetlany na ekranach telewizora.

Transmisja w trybie Półdupleks (Half-Dupleks) odbywa się wówczas gdy komunikujące się ze sobą urządzenia są zarówno nadajnikami jak i odbiornikami. Wymieniają one dane, ale nie robią tego w jednym czasie. Kiedy urządzenie A przesyła dane, urządzenie B może je tylko odbierać i nie może w tym samym czasie odpowiedzieć. Musi zaczekać, aż urządzenie A zakończy nadawanie i dopiero wówczas może odpowiedzieć i przesłać swoją porcję danych. Taki sposób przesyłu danych jest mało efektywny szczególnie, gdy zachodzi konieczność przerwania transmisji, a odbiornik nie może o tym poinformować nadajnika, gdyż musi zaczekać na zakończenie transmisji. Efektywny transfer danych w tym trybie jest niższy.

Transmisja w trybie Pełny Dupleks (Full-Dupleks) odbywa się wówczas, gdy oba urządzenia w jednym czasie mogą dane nadawać jak i odbierać. Dzięki temu istnieje pełna komunikacja pomiędzy urządzeniami zarówno w warstwie przesyłu danych jak i sterowania ich przepływem. Taki sposób również zwiększa efektywny transfer danych.

Symetrycznie i niesymetrycznie

Kolejnym określeniem, które występuje w dziedzinie transmisji danych, szczególnie łącz internetowych to pojęcie transmisji symetrycznej i niesymetrycznej. Jeżeli istnieje możliwość transmisji symetrycznej oznacza to, że zarówno odbieranie jak i wysyłanie danych może odbywać się z tą samą prędkością. W przypadku kiedy transmisja jest niesymetryczna wówczas odbieranie danych odbywa się z inną prędkością niż ich nadawanie. Typowym przykładem są łącza internetowe w różnych odmianach technologii DSL, które zazwyczaj umożliwiają szybsze odbieranie danych niż wysyłanie np. 10/1 Mb/s co oznacza, że maksymalna prędkość odbioru danych wynosi 10 Mb/s, zaś prędkość wysyłania danych 1 Mb/s.

W jaki sposób odbywa się komunikacja oraz przesył danych pomiędzy urządzeniami jest określone **protokołem komunikacji**.

Obecnie w nowoczesnych systemach komputerowych oraz dalekiego przesyłu danych wykorzystuje się szeregową transmisję ze względu na niższy koszt łączy. Nowoczesne technologie pozwalają na bardzo szybki przesył danych w technologii szeregowej, np. poprzez światłowody. Równoległy sposób przesyłu danych zanika i występuje w coraz mniejszej ilości rozwiązań, zazwyczaj tam, gdzie konieczna jest bardzo szybka wymiana dużej ilości danych na niewielkie odległości np. pomiędzy procesorem a pamięcią RAM.

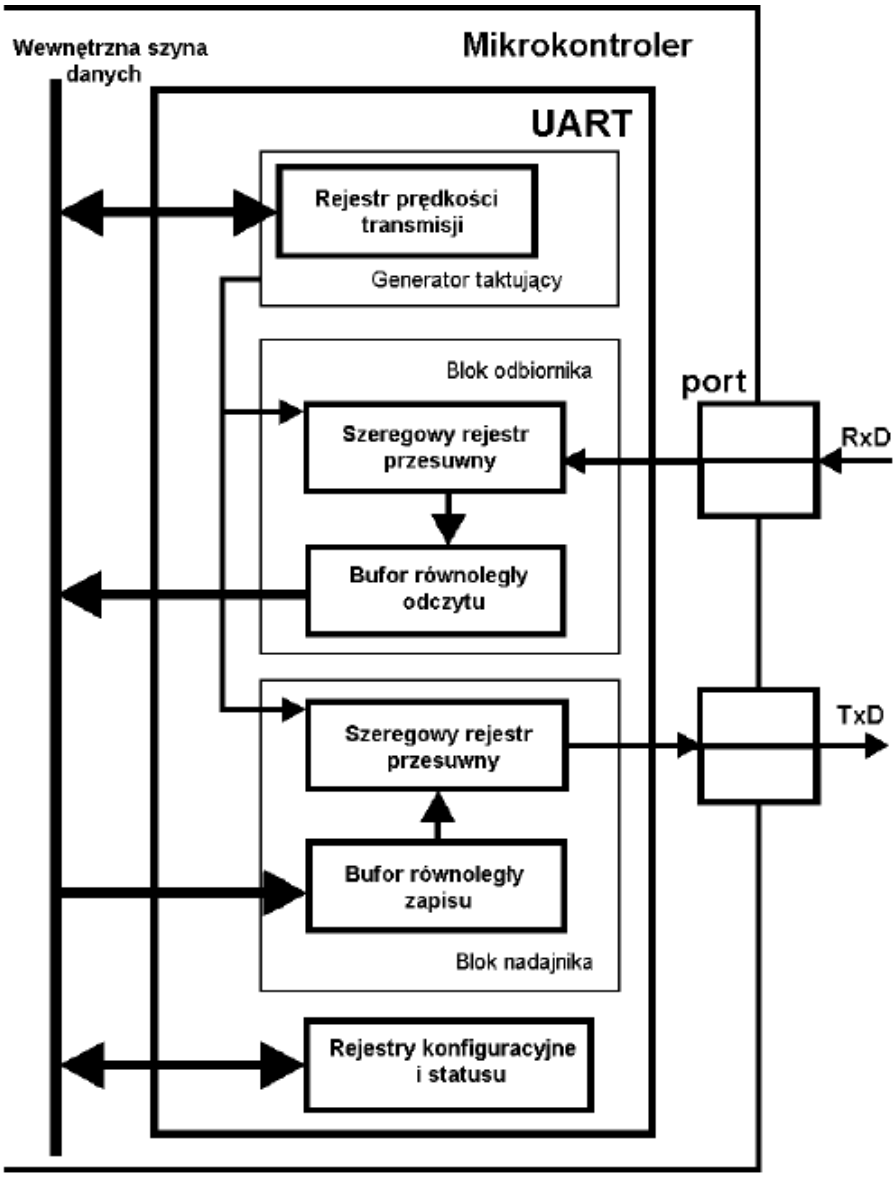
W mikrokontrolerach można wyróżnić następujące **interfejsy szeregowe**:

- **UART** (Universal Asynchronous Receiver/Transmiter),
- **SPI** (Serial Peripherial Interface)
- **1-Wire**
- **I^2C** (Inter-Integrated Circuit)
- **CAN** (Controller Area Network)
- **USB** (Universal Serial Bus)

USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter)

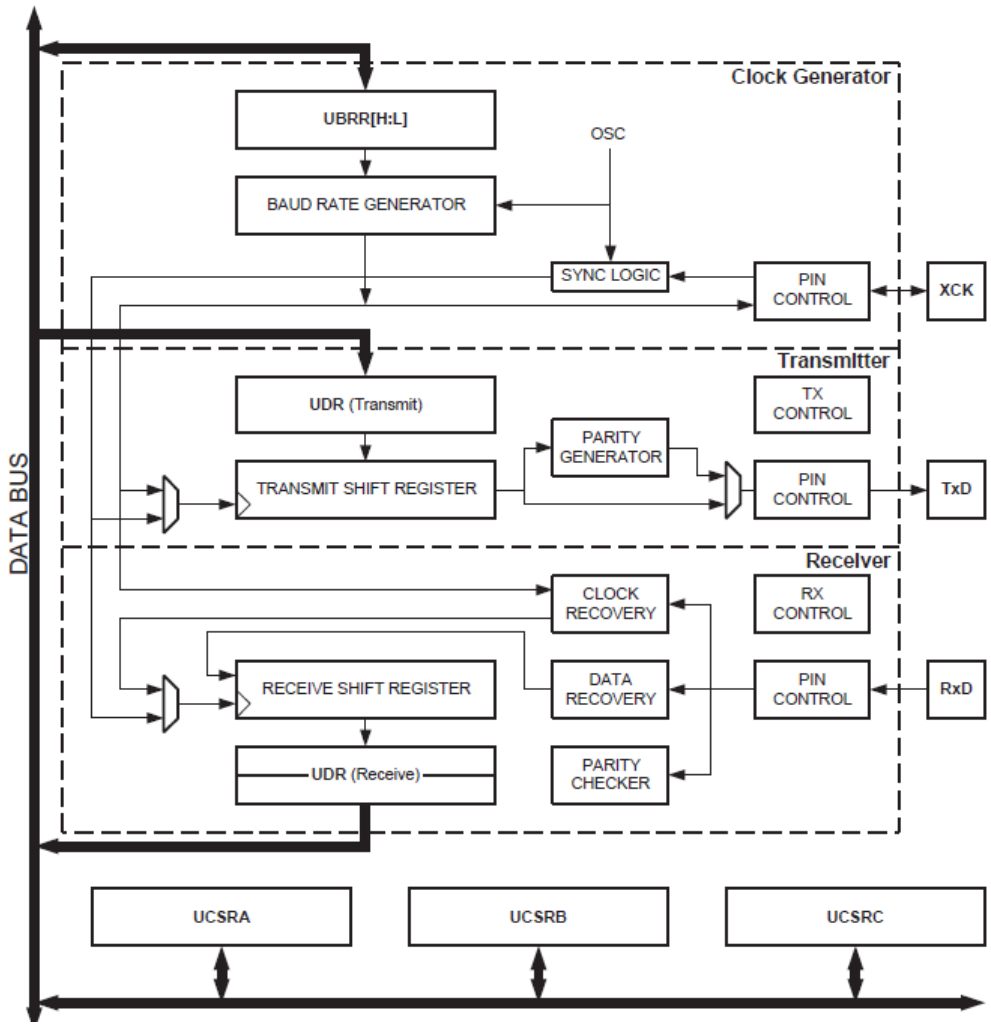
Uniwersalny synchroniczny i asynchroniczny odbiornik i nadajnik (USART) jest elastycznym urządzeniem do szeregowej transmisji danych. Jego główne cechy to:

- Pełny duplex, dwukierunkowa transmisja danych (niezależne rejestry szeregowego odbiornika i nadajnika)
- Operacje asynchroniczne i synchroniczne.
- Operacje synchroniczne taktowane zegarem Master-Slave
- Generator szybkiej transmisji o wysokiej rozdzielczości
- Wsparcie dla szeregowych ramek o 5,6,7,8, lub 9 bitach danych i 1 lub 2 bitach stopu
- Parzyste lub nieparzyste generatory parzystości i sprawdzanie parzystości wspierane sprzętowo
- Wykrycie przepełnienia danych
- Detekcja błędów ramki
- Filtrowanie zakłóceń zawierające wykrywanie fałszywych bitów startu i binarny filtr dolnoprzepustowy
- Trzy oddzielne przerwania: zakończenie nadawania, pusty rejestr danych TX, zakończenie odbioru
- Tryb komunikacji wieloprocesorowej
- Tryb komunikacji asynchronicznej o podwójnej prędkości



Schematyczna budowa interfejsu UART w mikrokontrolerze

USART Block Diagram[1]



Rejestry:

UBRR – szybkość transmisji

UDR (Transmit) – bufor danych nadawanych

UDR (Receive) – bufor danych odbieranych

Struktura układu

Układ składa się z trzech oddzielnych bloków:

− nadajnik,

− odbiornik,

− generator taktujący.

Zasada działania - **układ nadajnika**: Transmisja jest inicjowana przez wpis danej do rejestru danych UDR. Jeśli dana jest 9–bitowa, wcześniej należy wpisać wartość 9-tego bitu do rejestru UCSRB - bit TXB8. Dana jest przesyłana z rejestru UDR do rejestru przesuwnego gdy :

− nowa dana została wpisana do UDR po tym jak wysłany został ostatni bit danej poprzedniej. Rejestr przesuwny jest ładowany natychmiast.

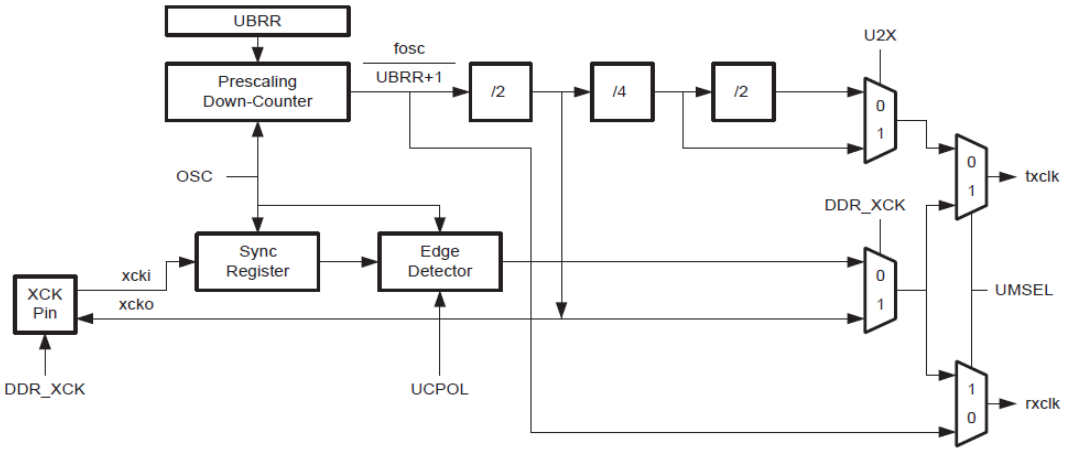
− nowa dana została wpisana do UDR zanim zakończono wysyłanie poprzedniej. Rejestr przesuwny jest ładowany dopiero po wysłaniu bitu stopu danej poprzedniej.

Zasada działania – **układ odbiornika**: Odbiornik próbkuje sygnał na wejściu RxD z częstotliwością 16*BAUD (BAUD – częstotliwość transmisji). Gdy linia jest w stanie spoczynkowym (IDLE – stan wysoki), pojedyncza próbka o stanie 0 jest interpretowana jako zbocze opadające bitu startu i rozpoczyna się procedura sprawdzająca - czy stan 0 na linii jest bitem startu czy zakłóceniem. W tym drugim przypadku zbocze jest ignorowane. Po stwierdzeniu poprawności bitu startu (3 kolejne próbki z rzędu mają wartość 0), następuje odczyt bitów danej, rozpoczynając od LSB. Każdy bit próbkowany jest 3 razy i jako prawdziwa wartość interpretowana jest ta, która pojawia się w przynajmniej 2 próbkach - eliminacja zakłóceń. Kiedy do odbiornika trafia bit stopu, przechodzi on procedurę sprawdzenia poprawności – jeśli z 3 próbek 2 lub 3 są zerami, ustawiana jest flaga FE (błąd ramki). Niezależnie od ewentualnego błędu ramki, ustawiana jest flaga RXC, a dana jest przesyłana do UDR. Jeśli ustawiony jest tryb odbioru danej 9–bitowej, bit RXB8 w rejestrze UCSRB jest ładowany wartością 9-tego bitu danej odebranej. UWAGA! Przed odczytaniem rejestru UDR należy więc zawsze sprawdzać stan flagi FE.

Ogólna obsługa interfejsu UART:

− wybór prędkości transmisji – wpisanie odpowiedniej wartość do rejestru prędkości,
− wybór 8 lub 9 bitowy format danych w rejestrze konfiguracyjnym,
− odblokowanie przerwania od nadajnika i/lub odbiornika (koniec wysyłania/odbioru danej),
− włączenie nadajnika i/lub odbiornika,
− uruchomienie nadawania
– wprowadzenie danej do rejestru danych zapisu UDR,
− odbiór
– odczyt danej z rejestru danych odczytu UDR, po sprawdzeniu flagi końca odbioru.

Figure 22-2. Clock Generation Logic, Block Diagram



Signal description:

txclk	Transmitter clock (Internal Signal).
rxclk	Receiver base clock (Internal Signal).
xcki	Input from XCK pin (internal Signal). Used for synchronous slave operation.
xcko	Clock output to XCK pin (Internal Signal). Used for synchronous master operation.
f_{osc}	XTAL pin frequency (System Clock).

Generator szybkości transmisji

Rejestr szybkości transmisji UBRR wraz z licznikiem liczącym w dół stanowi programowalny preskaler lub generator szybkości transmisji. Licznik pracuje z częstotliwością zegara systemowego. Do określenia szybkości transmisji konieczne jest odpowiednie dobranie wartości rejestru UBRR. Do wyliczenia tej wartości należy skorzystać z następującej zależności:

$$UBRR = \frac{f_{OSC}}{16\,BAUD} - 1$$

gdzie:

f_{OSC} – częstotliwość oscylatora [Hz] (pracy mikrokontrolera),

BAUD – pożądana wartość szybkości transmisji [bps]

Poniższa tabela zawiera zestawienie wartości UBRR dla kilku wybranych szybkości transmisji przy częstotliwości oscylatora z możliwie małym błędem (0,2 %).

Tabela . Zestawienie przykładowych wartości UBBR

BAUD	***UBRR*** (fOSC = 8MHz)	***UBRR*** (fOSC = 12MHz)
2400	207	312
4800	103	155
9600	51	77
19200	25	38

Inicjalizacja transmisji

Przed jakąkolwiek transmisją, moduł USART musi zostać zainicjalizowany. Inicjalizacja obejmuje ustawienie szybkości transmisji, wybór formatu ramki oraz odblokowanie układu odbiorczego i/lub nadawczego. Jeśli operacje USART mają być źródłem przerwań, to inicjalizacja powinna być wykonana z wykasowaną globalną flagą przerwań.

Rejestr UDR

Rejestr danych IO USART

Rejestr UDR

Bit	7	6	5	4	3	2	1	0
	RXB[7:0] (odczyt)							
	TXB[7:0] (zapis)							
Dostęp	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Wartość początkowa	0	0	0	0	0	0	0	0

Rejestr nadawczy i odbiorczy USART współdzielą ten sam adres przestrzeni adresowej IO. Zapis do rejestru spowoduje umieszczenie danych w buforze nadawczym (TXB), natomiast poprzez odczyt spod adresu tego rejestru spowoduje odczytanie danych rejestru odbiorczego (RXB).

Bufor nadawczy może być nadpisany tylko wówczas, gdy jest ustawiona flaga UDRE w rejestrze UCSRA, w innym przypadku próba wpisania jest ignorowana. Wpisanie danych do buforu nadawczego, podczas gdy moduł nadawczy (transmitter) jest odblokowany, spowoduje przeniesienie danych do rejestru przesuwnego (jeśli rejestr przesuwny jest pusty), a to jest równoznaczne z rozpoczęciem transmisji na pinie TXD.

Bufor odbiorczy składa się z dwóch buforów FIFO. Ze względu na budowę, nie należy korzystać z instrukcji bitowych pod adresem rejestru, gdyż zmienią one stan buforu.

Rejestr UCSRA

Rejestr A kontroli i statusu USART

Rejestr UCSRA

Bit	7	6	5	4	3	2	1	0
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM
Dostęp	R	R/W	R	R	R	R	R/W	R/W
Wartość_początkowa	0	0	1	0	0	0	0	0

RXC - Odbiór zakończony.
Flaga jest ustawiona, jeśli w buforze odbiorczym znajdują się nie odczytane dane. Flaga jest wykasowana, jeśli bufor jest pusty. Wyłączenie układu odbiornika kasuje zawartość buforu, więc i bit RXC jest także kasowany. Flaga RXC może generować przerwanie zakończenia odbioru (zob. opis RXCIE).

TXC - Transmisja zakończona.
Flaga jest ustawiana, gdy cała ramka zostanie wysłana z nadawczego rejestru przesuwnego, oraz bufor nadawczy jest pusty. Flaga jest automatycznie kasowana przy wykonywaniu instrukcji spod wektora przerwań związanego z zakończeniem transmisji, lub może być skasowana przez wpisanie jedynki.
Flaga TXC może generować przerwanie zakończenia transmisji (zob. opis bitu TXCIE).

UDRE - Pusty rejestr danych.

Flaga UDRE pojawia się jeśli bufor UDR jest gotowy na przyjęcie nowych danych. Flaga UDRE może generować przerwanie, gdy rejestr transmisji danych będzie pusty (zob. opis bitu UDRIE). Po resecie mikrokontrolera flaga jest ustawiana na 1.
FE - Błędna ramka.
Bit ten jest ustawiony jeśli nadchodząca ramka w buforze odbiorczym jest błędna np. bit stopu w nadchodzącej ramce ma wartość 0. Bit jest ważny do odczytania buforu odbiorczego UDR. Bit FE ma wartość 0, gdy bit stopu odebranych danych ma wartość 1. Zapisując do UCSRA należy zawsze przypisać temu bitowi wartość 0.

DOR - Przepełnienie danych.
Ten bit ma wartość 1, gdy zostało wykryte zdarzenie przepełnienia danych. Zdarzenie to następuje, gdy bufor odbiorczy jest pełny, a został wykryty kolejny bit startu. Bit jest ważny aż do odczytania buforu odbiorczego (UDR). Zapisując do UCSRA należy zawsze wpisywać temu bitowi wartość 0.

PE - Błąd parzystości.
Bit ten ma wartość 0, gdy zawartość rejestru odbiorczego ma błąd parzystości oraz jest włączona kontrola parzystości (UPM1=1). Bit traci ważność z chwilą odczytania rejestru odbiorczego. Zapisując do UCSRA należy wpisywać 0.

U2X - Podwójna szybkość transmisji.

Bit ma znaczenie tylko w transmisji asynchronicznej. Przy korzystaniu z trybu synchronicznego należy go zawsze zerować. Ustawienie bitu na 1 zmniejsza dzielnik szybkości transmisji z 16 na 8, co włącza tryb podwójnej szybkości dla transmisji asynchronicznej.

MPCM - Tryb komunikacji między procesorami.
Jeśli MPCM zostanie ustawiony na 1, wszystkie nadchodzące ramki odbierane przez odbiornik USART, które nie zawierają informacji o adresie, zostaną zignorowane. MPCM nie ma wpływu na pracę nadajnika.

Rejestr UCSRB

Rejestr B kontroli i statusu USART

Rejestr UCSRB

Bit	7	6	5	4	3	2	1	0
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
Dostęp	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Wartość_początkowa	0	0	0	0	0	0	0	0

RXCIE - Odblokowanie przerwania zakończenia odbioru.
Ustawienie tego bitu zezwala na generowanie przerwania na pojawienie się flagi RXC. Aby przerwanie zostało wygenerowane, muszą być ustawione flagi RXCIE oraz globalna flaga przerwań I.

TXCIE - Odblokowanie przerwania zakończenia nadawania.
Ustawienie tego bitu zezwala na generowanie przerwania na pojawienie się flagi TXC. Aby przerwanie zostało wygenerowane, muszą być ustawione flagi TXCIE oraz globalna flaga przerwań I.

UDRIE - Odblokowanie przerwania na pusty rejestr danych.
Ustawienie tego bitu zezwala na generowanie przerwania na pojawienie się flagi UDRE. Aby przerwanie zostało wygenerowane, muszą być ustawione flagi UDRIE oraz globalna flaga przerwań I.

RXEN - Odblokowanie odbiornika.
Ustawienie tego włącza odbiornik USART. Włączenie odbiornika powoduje, że funkcja RxD staje się nadrzędną nad normalnymi funkcjami IO tego pinu. Wyłączenie odbiornika czyści zawartość rejestru odbiorczego oraz unieważnia flagi FE, DOR i PE.

TXEN - Odblokowanie nadajnika.
Ustawienie tego włącza nadajnik USART. Włączenie nadajnika powoduje, że funkcja TxD staje się nadrzędną nad normalnymi funkcjami IO tego pinu. Wyłączanie nadajnika (wpisywanie 0 do TXEN) nie będzie dawało rezultatu, jeśli nie nastąpi zakończenie trwającego nadawania.

UCSZ2 - Rozmiar znaku.
Bit UCSZ2 wraz z bitami UCSZ1:0 stanowi kombinację, która konfiguruje liczbę bitów danych w ramce.

RXB8 – 8 bit danych odbieranych.

RXB8 stanowi 9 bit danych odbieranego znaku, jeśli operacje transmisji obywają się w ramkach o takim formacie. Odczyt tego bitu musi nastąpić przed odczytaniem pozostałej części z rejestru UDR.

TXB8 - 8 bit danych nadawanych.
TXB8 stanowi 9 bit danych znaku do nadania, jeśli operacje transmisji obywają się w ramkach o takim formacie. Ustawienie tego bitu musi nastąpić przed zapisem pozostałej części do rejestru UDR.

Rejestr UCSRC

Rejestr C kontroli i statusu USART

Rejestr UCSRC współdzieli ten sam adres przestrzeni adresowej IO wraz z rejestrem UBRRH. Zobacz opis bitu URSEL.

Rejestr UCSRC

Bit	7	6	5	4	3	2	1	0
	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
Dostęp	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Wartość_początkowa	0	0	0	0	0	0	0	0

URSEL - Wybór rejestru.
Bit ten służy do przełączania dostępu między rejestrami UBRRH i UCSRC. Ma wartość 1 jeśli odczytany został rejestr UCSRC. Chcąc zapisać do rejestru UCSRC należy do tego bitu wpisywać wartość 1. Porównaj z opisem dla rejestru UBRRH.

UMSEL - Wybór trybu USART.
Bit ten służy do wyboru między pracą synchroniczną a asynchroniczną. Do pracy asynchronicznej bit musi mieć wartość 0, natomiast do synchronicznej należy wpisać ustawić na 1.

UPM1:0 - Tryb parzystości.
Za pomocą tych bitów ustawia się rodzaj parzystości, która jest generowana przy nadawaniu oraz sprawdzana odbieraniu każdej ramki:

UPM1	UPM0	Tryb parzystości
0	0	Parzystość wyłączona
0	1	zastrzeżone
1	0	Włączony, tryb nieparzystości
1	1	Włączony, tryb parzystości

USBS - Wybór bitu stopu.
Ustawienie tych bitów decyduje o liczbie bitów stopu ustawianych przy nadawaniu ramki. Ustawienie to nie ma wpływu na pracę odbiornika.

USBS	Liczba bitów stopu
0	1 bit
1	2 bity

UCSZ1:0 - Liczba bitów danych w ramce.

Bit UCSZ2 (rejestr UCSZB) wraz z bitami UCSZ1:0 stanowi kombinację, która konfiguruje liczbę bitów danych w ramce i jest używana przez nadajnik i odbiornik.

UCSZ2	UCSZ1	UCSZ0	Rozmiar znaku
0	0	0	5 bitów
0	0	1	6 bitów
0	1	0	7 bitów
0	1	1	8 bitów
1	0	0	zastrzeżone
1	0	1	zastrzeżone
1	1	0	zastrzeżone
1	1	1	9 bitów

UCPOL - Polaryzacja zegara.
Bit ten jest używany tylko w trybie pracy synchronicznej. Przy pracy asynchronicznej należy go ustawiać na wartość 0.

Rejestry UBRRL i UBRRH

Rejestr szybkości transmisji

Rejestr UBRRH współdzieli ten sam adres przestrzeni adresowej IO wraz z rejestrem UCSRC. Zobacz opis bitu URSEL.

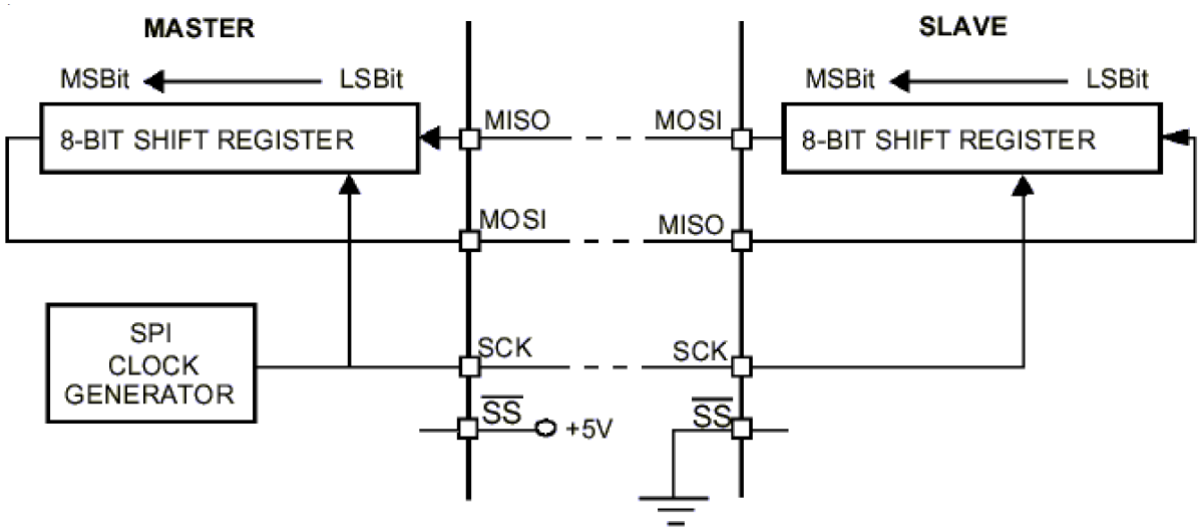
Rejestr UBRR

Bit	7	6	5	4	3	2	1	0
UBRRH	URSEL	-	-	-	UBRR[11:8]			
UBRRL	UBRR[7:0]							
Dostęp	R/W	R	R	R	R/W	R/W	R/W	R/W
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Wartość początkowa	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0

URSEL - Wybór rejestru.
Bit ten służy do przełączania dostępu między rejestrami UBRRH i UCSRC. Ma wartość 0 jeśli odczytany został rejestr UBRRH. Zapisując do rejestru UBRRH należy do tego bitu wpisywać wartość 0. Porównaj z opisem dla rejestru UCSRC.
Bity 6:4 - Zarezerwowane do przyszłych zastosowań, zapisując ustawiać na wartość 0.
UBRR11:0 - Rejestr szybkości transmisji.
Te bity stanowią 12 bitowy rejestr wyboru szybkości transmisji modułu USART. UBRRH zawiera 4 najbardziej znaczące bity, natomiast UBRRL 8 mniej znaczących. Nadpisanie bitów UBRR wprowadza natychmiastową zmianę preskalera w generatorze szybkości transmisji. Jak obliczać wartość UBRR napisano w punkcie „Generator szybkości transmisji”.

Interfejs SPI

Interfejs SPI służy do **dwukierunkowej** (*full-duplex*), **synchronicznej**, szeregowej transmisji danych pomiędzy mk, a zewnętrznymi układami peryferyjnymi, np. przetwornikami A/C i C/A, szeregowymi pamięciami EEPROM, innymi mk, potencjometrami cyfrowymi, programowalnymi generatorami, itd. Jest interfejsem **trójżyłowym**: składa się z **dwóch linii synchronicznie przesyłających dane** w przeciwnych kierunkach i **linii z sygnałem zegarowym** synchronizującym ten transfer. Dodatkowo mk wyposażone są w czwartą linię SS służącą do wyboru trybu pracy interfejsu SS = 1 – układ master, SS = 0 – układ slave.



Na rysunku pokazano sposób połączenia interfejsu SPI pomiędzy **układem nadrzędnym** (*master*) i **układem podrzędnym** (*slave*), gdzie pin MISO jest wejściem danych, pin MOSI wyjściem danych, a pin SCK wyjściem zegara dla układu master i wejściem zegara dla układu slave. Z rysunku widać, że protokół transmisji (w najprostszej postaci) wymaga dwóch rejestrów przesuwnych (*shift register*) połączonych w **licznik pierścieniowy** (wejście jest połączone z wyjściem). Transmisja jest synchroniczna - jeden z układów dostarcza sygnału zegara, odseparowanego od sygnału danych, zgodnie ze zboczami zegara taktującego. **Układ generujący sygnał zegara jest określony jako nadrzędny**, bez względu na to czy dane są przez niego nadawane czy odbierane. Wszystkie pozostałe układy na magistrali są określone jako **podrzędne**. Sygnały danych i zegara są przesyłane oddzielnymi jednokierunkowymi liniami. Sygnał zegara nie jest ciągły, nadawany jest jedynie w czasie trwania transmisji. Umożliwia to odbiór poszczególnych bitów danych bez konieczności testowania bitów startu i stopu, charakterystycznego dla transmisji asynchronicznej.

Kiedy układ nadrzędny nadaje dane na linii MOSI do układu podrzędnego, to układ podrzędny odpowiada, wysyłając do układu nadrzędnego dane na linii MISO. Implikuje to dwukierunkową transmisję z jednoczesnym wysyłaniem i odbieraniem danych, synchronizowanym tym samym sygnałem zegarowym, przez oba układy. Zatem bajt transmitowany jest od razu zastępowany bajtem odbieranym. Dzięki temu nie ma „pustych" transmisji: raz aby wysłać bajt, drugi aby go odebrać. Do wysłania bajtu i jego odebrania wystarczy osiem impulsów zegarowych na linii SCK.

W tym interfejsie dane są wpisywane (odbierane) do rejestru szeregowego jednym zboczem zegarowym, a przesuwane (wyprowadzane na zewnątrz – nadawane) drugim. Zwiększa to czas podtrzymania danych odbiornika do ½TCLK - td, gdzie TCLK - okres zegara, td -

opóźnienie magistrali. Zatem wypełnienie sygnału zegarowego wynosi około ½. Najczęściej szybkość transmisji dla mk nie przekracza 1-2 Mbit/s.
Wymiana danych za pomocą interfejsu SPI mk pracującego w trybie master pomiędzy mk, a zewnętrznym urządzeniem peryferyjnym generalnie przebiega według następującej procedury:

- najpierw należy odpowiednio skonfigurować interfejs SPI mk: tryb master, poprawnie skonfigurowane piny interfejsu (pin SCK ma być wyjściem), ustawić częstotliwość sygnału zegarowego,
- urządzenie peryferyjne musi być uaktywnione (przygotowane na odbiór danych) – najczęściej służy do tego dodatkowa linia mk podłączona do wejścia CS (*Chip Select*) urządzenia peryferyjnego,
- aby rozpocząć transmisję wpisuje się daną do rejestru przesuwnego,
- po czym czeka się na zakończenie transmisji, testując flagę informującą o zakończeniu transmisji lub czeka się na przerwanie od układu SPI, o ile jest odblokowane,
- na zakończenie z rejestru przesuwnego można odczytać daną odebraną.

Interfejs SPI mk można skonfigurować do pracy w trybie slave. W tym przypadku procedura postępowania jest następująca:

- najpierw należy odpowiednio skonfigurować interfejs SPI mk: tryb slave, poprawnie skonfigurowane piny interfejsu (pin SCK ma być wejściem),
- następnie wpisuje się daną, którą chcemy wysłać do rejestru przesuwnego,
- ustawia się odpowiedni bit w rejestrze sterującym SPI uruchamiającym interfejs,
- urządzenie peryferyjne musi być aktywne i pracować w trybie master,
- po czym czeka się na zakończenie transmisji, które wywołuje przerwanie, o ile jest odblokowane,
- na zakończenie z rejestru przesuwnego można odczytać daną odebraną.

W trybie slave metoda odpytywania flagi zakończenia transmisji jest nieefektywna, ponieważ transmisja z układu peryferyjnego może nastąpić w dowolnej chwili nieznanej mk. Zatem musi on w pętli głównej programu ciągle odpytywać tę flagę, co niepotrzebnie zajmuje czas procesora. Stąd wiele układów SPI posiada dodatkowy sygnał wejściowy (pin SS), który pozwala na wymuszenie trybu slave interfejsu SPI mk przez urządzenie peryferyjne.

Podsumowując, właściwości układu SPI z punktu widzenia interfejsu mogą być opisane przy pomocy 3 funkcji interfejsowych:

- **Talker** (nadajnik) – wprowadzenie danej (bajta) z wewnętrznej magistrali równoległej do rejestru szeregowego, przesuwanie i wysyłanie bit po bicie danej z rejestru szeregowego.
- **Listener** (odbiornik) – odczyt bit po bicie danej odbieranej i kompletowanie jej w rejestrze szeregowym, następnie po zebraniu całego bajta równoległy jego odczyt przez wewnętrzną magistralę mk.
- **Repeater** (pośredniczenie w transmisji) - dane wejściowe są bit po bicie wpisywane do rejestru szeregowego z linii wejściowej, przesuwane i od razu bit po bicie wysyłane na linię wyjściową.

Maksymalna konfiguracja zawiera z reguły wszystkie trzy wymienione funkcje, ale w praktyce spotyka się najczęściej dwie pierwsze. Trzecia funkcja pozwala na realizację interfejsowych systemów pętlowych.

Aby transmisja pomiędzy mk, a urządzeniem peryferyjnym przebiegała prawidłowo muszą być spełnione następujące warunki:

- zachowanie **jednakowej długo**ść danej (najczęściej 8 bitów lub wielokrotność tej liczby),
- **taka sama kolejno**ść **wysyłania bitów** (najczęściej od MSB do LSB, niektóre mk mają możliwość programowej zmiany tej kolejności),
- zgodna **polaryzacja i faza sygnału zegarowego**.

Częstotliwość sygnału zegarowego nie jest istotna, ponieważ dane są synchronizowane tym sygnałem i są aktywowane (wpisywane lub wyprowadzane z rejestru szeregowego) wyłącznie przez zbocza tego sygnału. Zatem prędkość transmisji możemy zmienić nawet w trakcie przesyłania danych. Można również wstrzymać transmisję w dowolnej chwili, aby następnie ją dalej kontynuować. Właściwość ta jest szczególnie przydatna, w przypadku przesyłania danych 8*n*-bitowych, gdzie *n* – liczna naturalna, za pomocą interfejsu SPI mk, który wyłącznie może przesyłać dane 8-bitowe. Po wysłaniu 8 bitów następuje przerwa – zatrzymanie transmisji, w czasie której do interfejsu jest wprowadzana/czytana kolejna dana 8-bitowa, po tej przerwie ponownie uruchamia się interfejs, itd.
Polaryzacja i faza sygnału zegarowego muszą być zgodne zarówno dla układu master, jak i układów slave, aby transmisja pomiędzy nimi przebiegała prawidłowo.
Polaryzacja sygnału zegarowego jest określona przez wartość logiczną sygnału zegara w stanie spoczynkowym (poza czasem transmisji):

- **CPOL=0** – sygnał zegara w stanie spoczynku jest w **stanie niskim** „Lo”,
- **CPOL=1** – sygnał zegara w stanie spoczynku jest w **stanie wysokim** „Hi”.

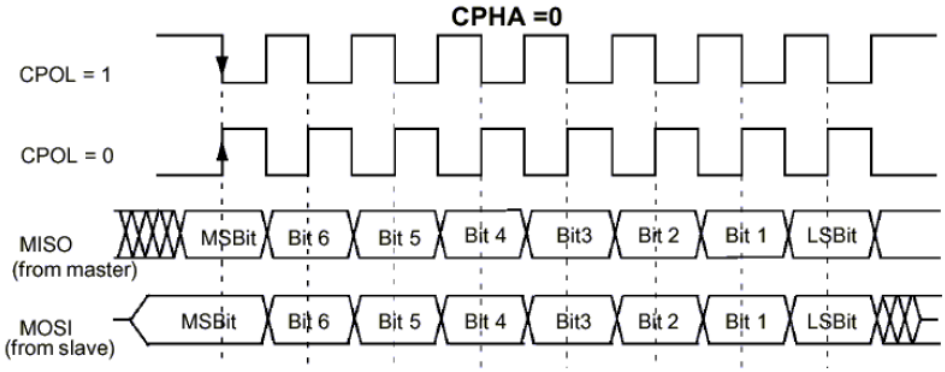
Faza sygnału zegarowego definiuje zależność pomiędzy zboczami sygnału zegarowego, a momentami próbkowania danych wejściowych; przesuwania zawartości rejestru (wysłania danych wyjściowych):

- **CPHA=0** – **pierwsze zbocze** sygnału zegarowego **próbkuje dane wejściowe**, drugie zbocze przesuwa dane w rejestrze – wyprowadza je z rejestru (dane są próbkowane, a następnie przesuwane i wysyłane),
- **CPHA=1** – **pierwsze zbocze** sygnału zegarowego **przesuwa dane** w rejestrze – wyprowadza je z rejestru, drugie zbocze próbkuje dane wejściowe (dane są przesuwane i wysyłane, a następnie próbkowane i wpisywane do rejestru).

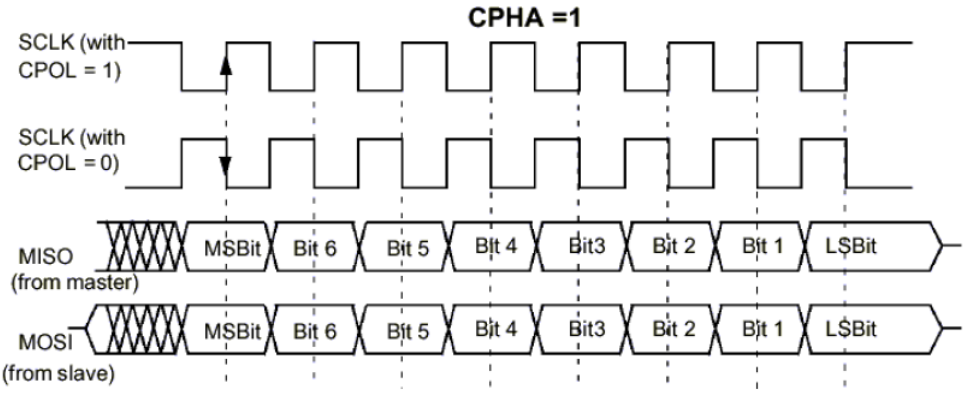
Stąd można wyróżnić cztery warianty pracy interfejsu SPI (strobowania/przesuwania informacji):

- (1): CPHA=0 i CPOL=0,
- (2): CPHA=0 i CPOL=1,
- (3): CPHA=1 i CPOL=0,
- (4): CPHA=1 i CPOL=1.

Najczęściej spotyka się pierwszy wariant, lecz pozostałe również są stosowane, dlatego we współczesnych mk istnieje możliwość programowego ustawienia polaryzacji i fazy sygnału zegarowego.



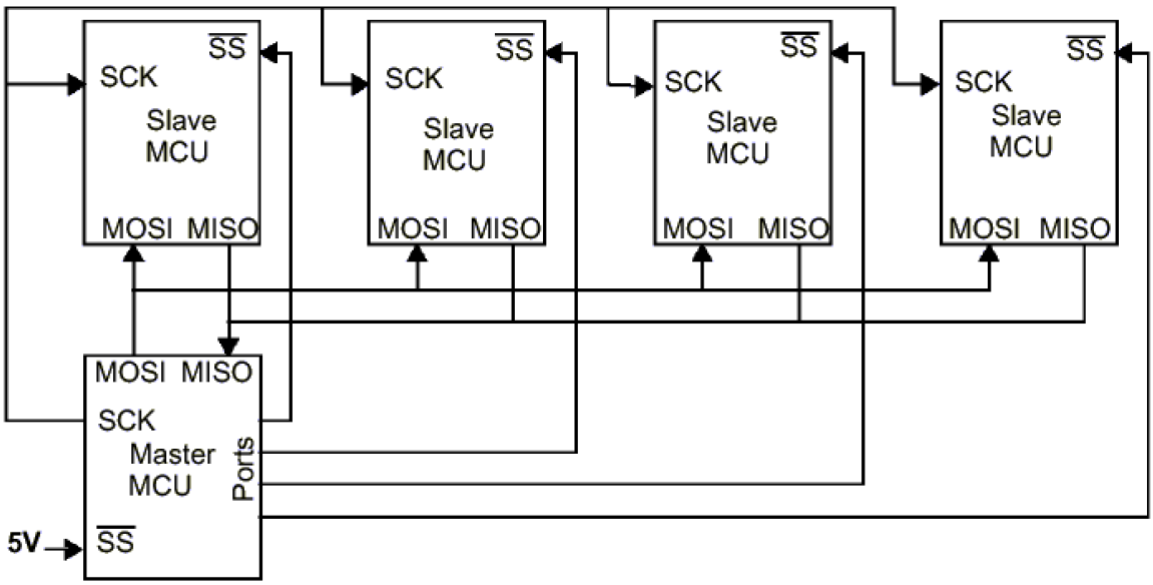
Rys. 2.61. Przebiegi czasowe interfejsu SPI dla sygnału zegarowego o CPHA=0



Rys. 2.62. Przebiegi czasowe interfejsu SPI dla sygnału zegarowego o CPHA=1

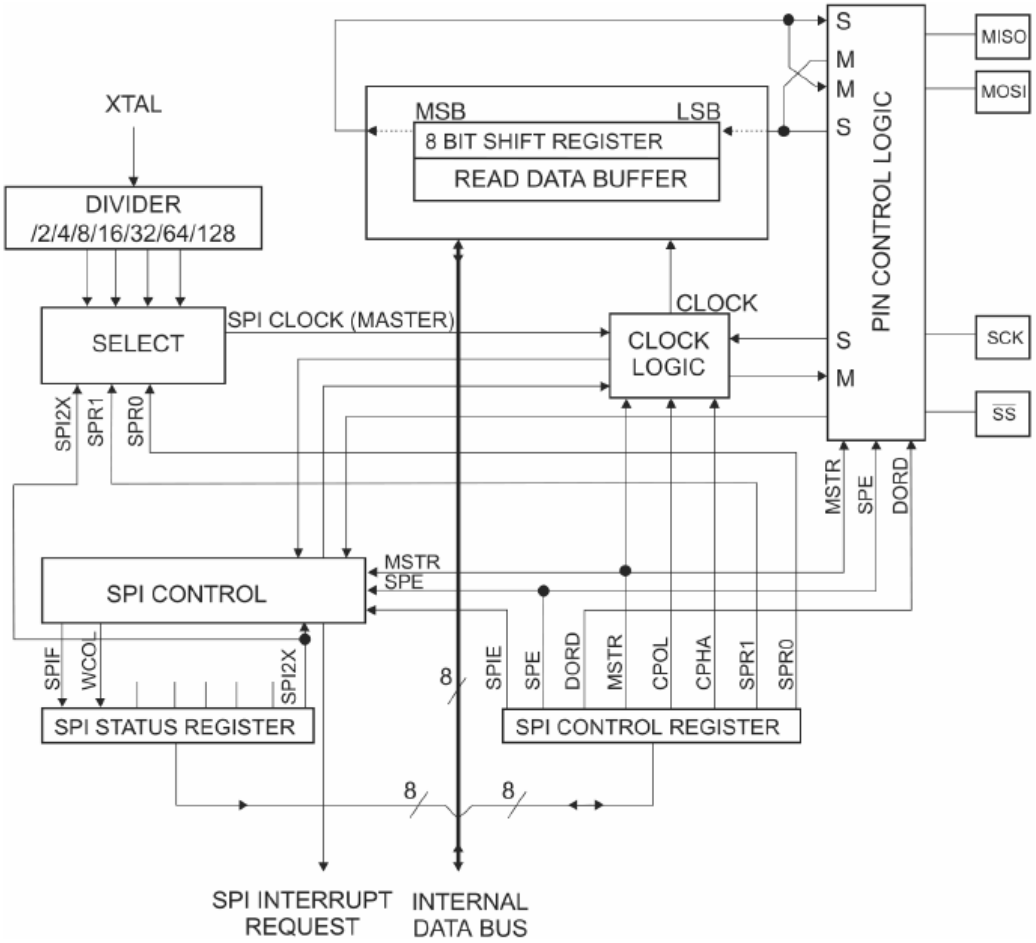
Na rys. 2.61 przedstawiono (1) i (2) wariant pracy interfejsu SPI, dla sygnału zegarowego o fazie CPHA=0, natomiast na rys. 2.62 warianty (3) i (4) dla sygnału zegarowego o fazie CPHA=1.

Na rys. 2.64 pokazano interfejsowy system magistralowy z pojedynczym układem master (*single master system*). Jest to często spotykana i zarazem najprostsza konfiguracja oparta na interfejsie SPI.



Rys. 2.64. System z pojedynczym układem master oparty na interfejsie SPI

Układ master wybiera poszczególny układ slave korzystając z czterech linii portu równoległego. Dane urządzenie slave jest wybrane, gdy na jego wejściu SS pojawi się stan niski. W czasie transmisji, w celu uniknięcia kolizji na liniach interfejsu, tylko jeden układ slave może być aktywny.



Schemat blokowy modułu SPI

Rejestr SPCR

Rejestr kontroli SPI

Rejestr SPCR

Bit	7	6	5	4	3	2	1	0
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
Dostęp	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Wartość_początkowa	0	0	0	0	0	0	0	0

SPIE - Odblokowanie przerwania SPI.

SPE - Bit włączenia interfejsu SPI.
Ustawienie tego bitu na 1 powoduje odblokowanie / włączenie SPI.

DORD - Kolejność danych.
Kiedy bit DORD jest ustawiony na 1, dane przesyłane są od najmniej znaczącego bitu (LSB). DORD = 0 oznacza wysyłanie od bitu najbardziej znaczącego (MSB).

MSTR - Wybór master/slave.
Przez wpisanie 1, tym bitem włącza się tryb master (rolę układu jako nadrzędnego). SPI będzie pracować jako slave jeśli ten bit będzie miał wartość logicznego zera. Do pracy układu jako master, pin powinien być wcześniej ustawiony jako wyjściowy.

CPOL - Polaryzacja zegara.
Jeśli do CPOL jest wpisane 1, to SCK w spoczynku jest w stanie wysokim, jeśli 0, to w stanie niskim.

CPHA - Faza zegara.
CPHA=1 – pierwsze zbocze sygnału zegarowego próbkuje dane wejściowe a drugie przesuwa dane w rejestrze, CPHA=0 – pierwsze zbocze przesuwa dane w rejestrze a drugie próbkuje dane wejściowe.

SPR1:0 - Szybkość zegara SPI.
Bity kontrolujące szybkość taktowania zegara SCK dla urządzenia skonfigurowanego jako Master (nadrzędny), w stosunku do częstotliwości oscylatora mikrokontrolera. Dla pracy jako slave, ustawienie nie ma znaczenia.

SPI2X	SPR1	SPR0	Częstotliwość SCK
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

Rejestr SPSR

Rejestr statusowy SPI

Rejestr SPSR

Bit	7	6	5	4	3	2	1	0
	SPIF	WCOL	-	-	-	-	-	SPI2X
Dostęp	R	R	R	R	R	R	R	R/W
Wartość_początkowa	0	0	0	0	0	0	0	0

SPIF - Flaga przerwania SPI.
Flaga jest ustawiana po zakończeniu transmisji. Przerwanie jest generowane jeśli ustawiony jest bit SPIE w rejestrze SPCR oraz ustawiona globalna flaga przerwań I. Flaga jest ustawiana także, gdy pin jest ustawiony jako wejściowy i w pojawi się na nim stan niski, a SPI ustawiony jest jako master. Flaga jest sprzętowo kasowana przy wykonywaniu programu spod wektora przerwań SPI. Flaga kasowana sprzętowo jest także, gdy po odczytaniu rejestru SPSI, nastąpi dostęp (odczyt lub zapis) do rejestru danych SPDR.

WCOL - Flaga kolizji zapisu.
Bit jest ustawiony, gdy nastąpi zapis do SPDR podczas trwającej transmisji. Flaga jest kasowana gdy po odczytaniu rejestru SPSI nastąpi dostęp do rejestru danych SPDR.

SPI2X - Podwójna szybkość SPI.
Ustawienie tego bitu na 1 podwaja szybkość pracy SPI (częstotliwość SCK).
Zobacz też opis bitów SPR1:0 w rejestrze SPCR.

Rejestr SPDR

Rejestr danych SPI

Rejestr danych SPI służy do odczytu i zapisu transmitowanych danych. Zapis do rejestru inicjuje transmisję. Odczytywanie rejestru powoduje odczyt buforu odbiorczego rejestru przesuwnego.

Rejestr SPSR

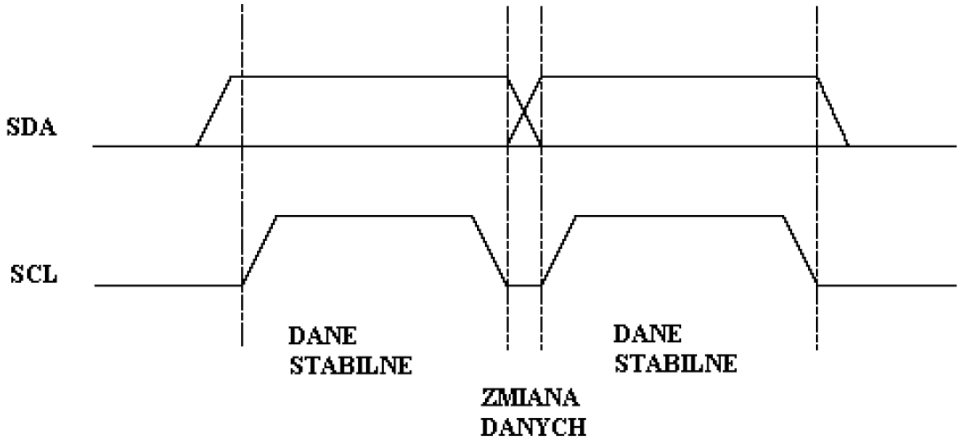
Bit	7	6	5	4	3	2	1	0
	MSB							LSB
Dostęp	R	R	R	R	R	R	R	R/W
Wartość_początkowa	0	0	0	0	0	0	0	0

Interfejs TWI (I^2C)

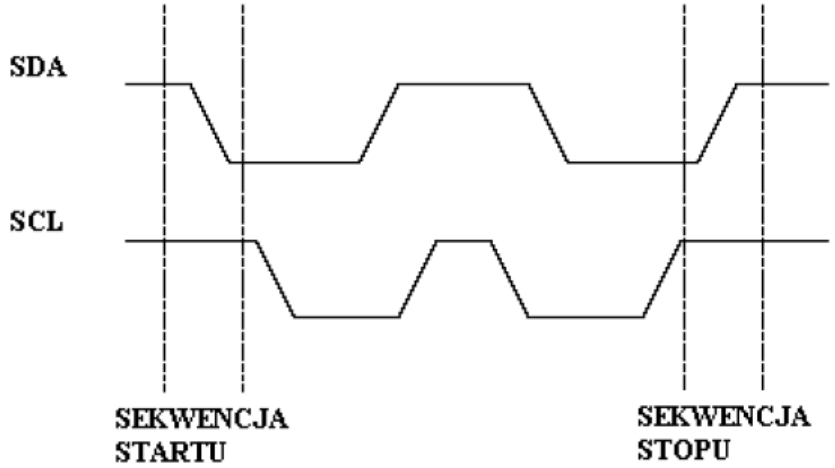
Interfejs ten jest jednym ze standardów w dziedzinie synchronicznych interfejsów szeregowych małego zasięgu. Służy do łączenia wielu nadajników-odbiorników do jednej, dwuprzewodowej magistrali szeregowej. Dołączane układy mogą pracować jako nadrzędne (sterować transmisją) jak i podrzędne (dopuszczalna jest przy tym jednoczesna obecność wielu układów obu rodzajów).
Interfejs TWI (two wire interface) jest w pełni zgodny z popularnym I2C firmy Philips, a Atmel używa innej nazwy aby nie łamać praw patentowych.

Interfejs I^2C podobnie jak SPI, został stworzony do komunikacji między układami scalonymi w obrębie jednego urządzenia. Komunikacja odbywa się dwuprzewodowo, linią danych SDA oraz linią zegara SCL, które mogą być wspólne dla wszystkich układów w systemie. Pomimo jednej linii danych, transmisja może odbywać się dwukierunkowo, jednak ze stosunkowo niewielką szybkością. W większości przypadków spotyka się możliwość pracy przy częstotliwości linii SCL do 100kHz, a rzadziej do 400kHz. Moduł TWI wbudowany w ATmega umożliwia pracę w obu tych trybach.
Współpraca wielu układów po tych samych przewodach jest możliwa dzięki temu, iż każdemu z urządzeń przypisuje się indywidualny adres. Liczba urządzeń podłączonych jednocześnie jest ograniczona jedynie maksymalną pojemnością linii 400pF oraz 7-bitową przestrzenią adresową dla urządzeń podrzędnych. Zasięg magistrali wynosi 1-100 m, zależnie od szybkości transmisji. Projektując magistralę TWI (I2C) nie należy zapomnieć o rezystorach podciągających.
Moduł TWI może pracować zarówno jako układ nadrzędny, jak i podrzędny. W trybie nadrzędnym sprawuje on nadzór nad magistralą, generując jednocześnie sygnał taktujący na linii SCL. TWI może wówczas wywoływać układy podrzędne przesyłając ich adres, a następnie wysyłać lub odbierać od nich dane. Jeśli do danej magistrali podłączone są dwa układy działające w trybie nadrzędnym, TWI pozwala na arbitrażowe rozstrzyganie transmisji jednoczesnych.

Stabilność i zmiana danych W stanie bezczynnym „idle” obie linie SCL i SDA znajdują się w stanie logicznym wysokim. Stan linii SDA może ulegać zmianom jedynie przy niskim stanie sygnału zegarowego, stan linii SDA musi być stabilny przy wysokim stanie linii SCL (rys.33). Naruszenie tej zasady w dwóch wyjątkowych sytuacjach pozwala na określenie początku i końca transmisji.



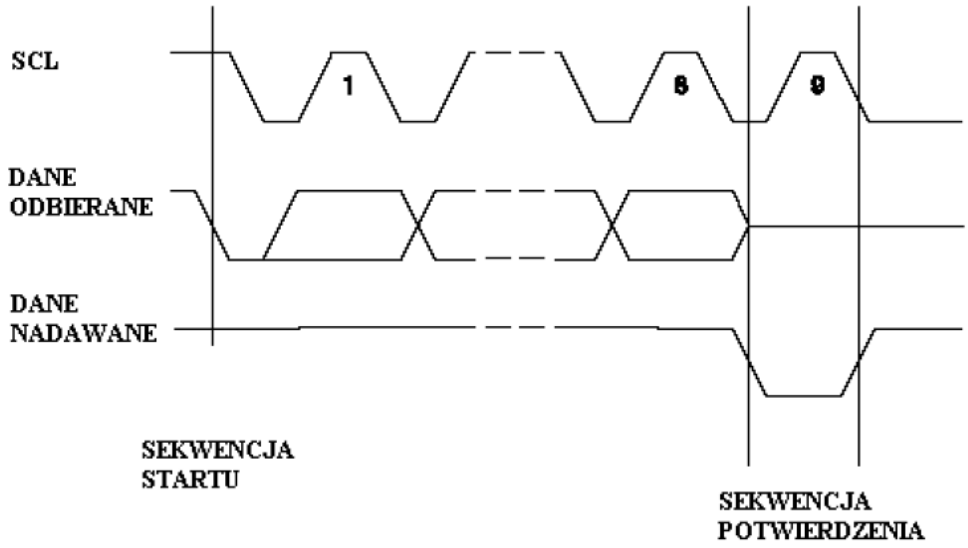
Rys.33. Impulsy na liniach SCL i SDA w trakcie transmisji danych [9] Sekwencja startu i stopu Zmiana stanu na linii SDA z wysokiego na niski podczas wysokiego poziomu na linii SCL oznacza sekwencję startu (rys.34), zaś zmiana w kierunku przeciwnym – sekwencję stopu (rys.34). Sekwencje startu i stopu generuje urządzenie MASTER.



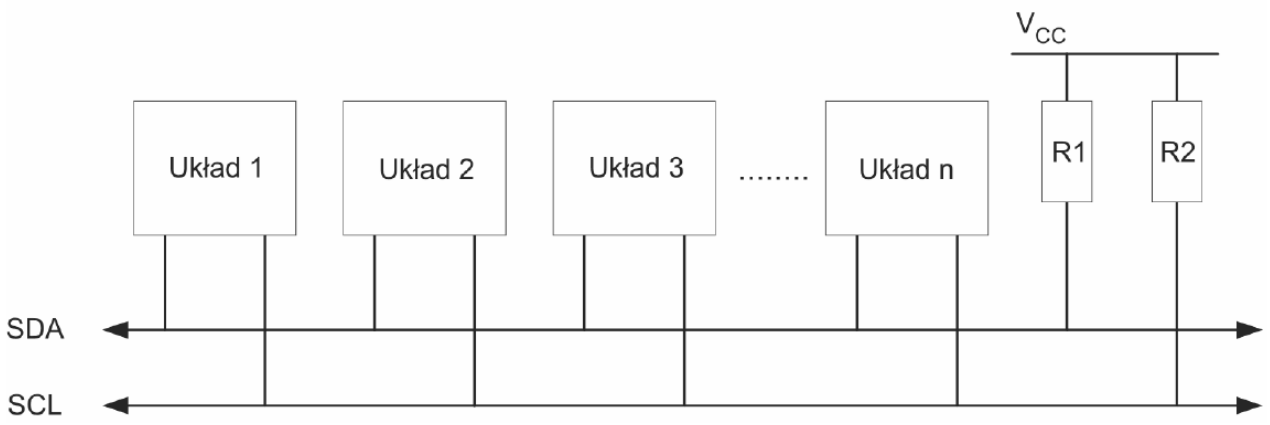
Sesja transmisji danych

Sesja połączeniowa inicjowana jest przez urządzenie typu MASTER przez wymuszenie stanu niskiego na linii SDA i wygenerowanie w ten sposób sekwencji startu. Następnie MASTER przesyła linią SDA osiem bitów słowa adresowego. Ósmy bit wybiera rodzaj operacji: logiczne 0 dla zapisu, logiczna 1 dla odczytu danych. Zaadresowane urządzenie SLAVE wymusza stan niski na zwolnionej przez MASTER linii SDA w trakcie dziewiątego impulsu zegarowego, co oznacza potwierdzenie odebrania słowa adresowego.

Dalsza transmisja danych przebiega podobnie. Urządzenie nadające przesyła osiem bitów danych do urządzenia odbierającego synchronicznie z 8-ma impulsami zegarowymi na linii SCL, a w czasie 9-tego impulsu SCL (rys.35) oczekuje na potwierdzenie w postaci logicznego 0. Sekwencja stopu kończy sesję transmisji danych.



Sekwencja potwierdzenia odbioru znaku[9]



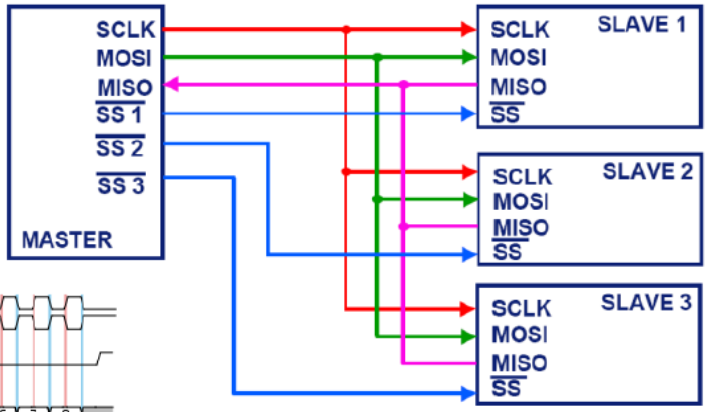
Schemat połączeń w magistrali TWI (I^2C)

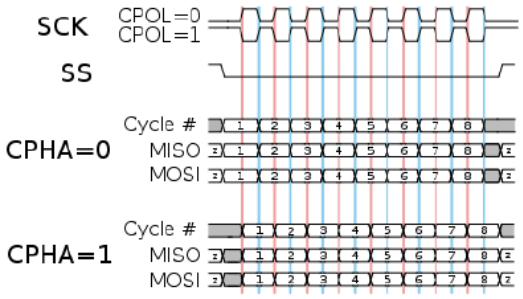
Interfejs SPI

- **SPI** (ang. *Serial Peripheral Interface, Motorola*) – synchroniczny, full dulex, szeregowy interfejs urządzeń peryferyjnych.
- Jeden z najczęściej używanych interfejsów komunikacyjnych pomiędzy systemami mikroprocesorowymi a układami peryferyjnymi takimi jak: przetworniki ADC/DAC, układy RTC, pamięci EEPROM, akcelerometry, żyroskopy, magnetometry, pamięci flash, karty MMC/SD/ itp.
- Komunikacja odbywa się synchronicznie za pomocą 4linii:
  - *MOSI* - (ang. Master Output Slave Input) - dane dla układu peryferyjnego
  - *MISO* - (ang. Master Input Slave Output) - dane z układu peryferyjnego
  - *SCK* - (ang. Serial Clock) - sygnał zegarowy (taktujący)
  - Do aktywacji wybranego układu peryferyjnego służy dodatkowo linia *CS* (ang. *Chip Select* - wybór układu).
- Opierając się na założeniach SPI, istnieją interfejsy typu QSPI, Microwire, 3-wire serial bus, Multi I/O SPI

Interfejs typu SPI

Połączenie trzech SLAVE-ów do Mastera w SPI

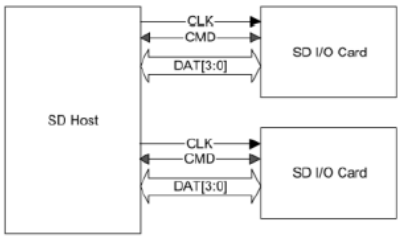


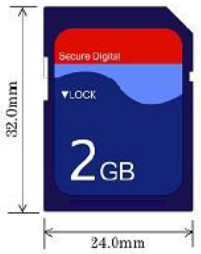


Cztery możliwe tryby pracy interfejsu SPI

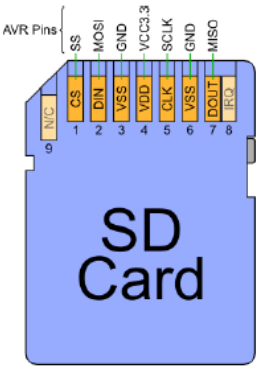
Synchroniczny interfejs typu SPI – 4-przewodowy (sygnały SCLK, MOSI, MISO i sygnał wyboru układu /SS),

Specyfikacja karty pamięci typu SD





Pin	SD 4-bit mode		SD 1-bit mode		SPI mode	
1	CD/DAT[3]	Data line 3	N/C	Not Used	CS	Card Select
2	CMD	Command line	CMD	Command line	DI	Data input
3	VSS1	Ground	VSS1	Ground	VSS1	Ground
4	VDD	Supply voltage	VDD	Supply voltage	VDD	Supply voltage
5	CLK	Clock	CLK	Clock	SCLK	Clock
6	VSS2	Ground	VSS2	Ground	VSS2	Ground
7	DAT[0]	Data line 0	DATA	Data line	DO	Data output
8	DAT[1]	Data line 1 or Interrupt (optional)	IRQ	Interrupt	IRQ	Interrupt
9	DAT[2]	Data line 2 or Read Wait (optional)	RW	Read Wait (optional)	NC	Not Used



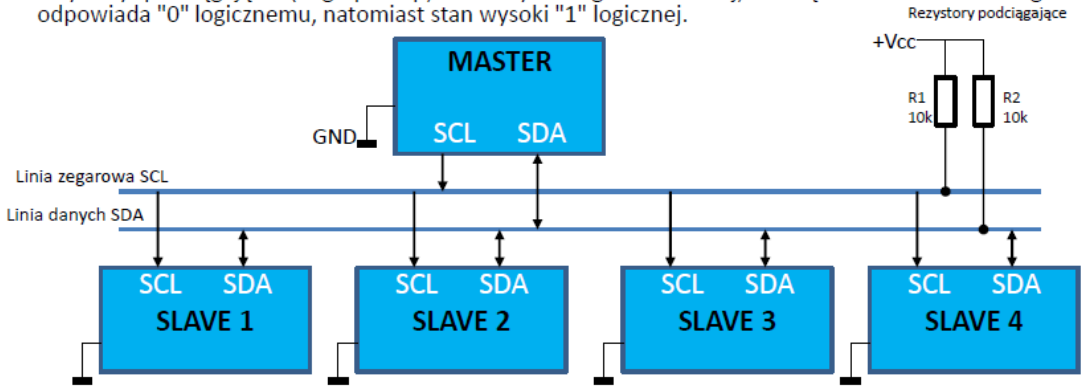
Zasilanie karty od 2.0V do 3.6V

Interfejs I2C

- **I²C** – szeregowa, dwukierunkowa magistrala służąca do przesyłania danych w urządzeniach elektronicznych. Została opracowana przez firmę Philips na początku lat 80.
- Znana również pod akronimem IIC, którego angielskie rozwinięcie Inter-Intergrated Circuit oznacza "pośredniczący pomiędzy układami scalonymi".
- Standard I²C określa dwie najniższe warstwy modelu odniesienia OSI: warstwę fizyczną i warstwę łącza danych.
- Standard został opracowany na początku lat 80. (określany obecnie jako tryb standardowy pracy) i cechowały go:
  - prędkość transmisji 100 kbps
  - 7-bitowa przestrzeń adresowa
  - W 1992 roku została opracowana wersja 1.0 standardu, która wprowadzała następujące zmiany:
  - dodanie trybu pracy z prędkością transmisji 400 kbps (Fast Mode)
  - rozszerzenie standardu o możliwość adresowania 10-bitowego
  - W 1998 roku opracowana została wersja 2.0:
  - dodanie trybu High Speed Mode, pozwalającego na prędkość transmisji 3,4 Mbps
  - Zwiększenie zakresu tolerancji napięcia w stanie wysokim: 2,3 – 5,5 V
  - W 2000 roku powstała wersja 2.1, wprowadzająca drobne zmiany.

Interfejs I2C

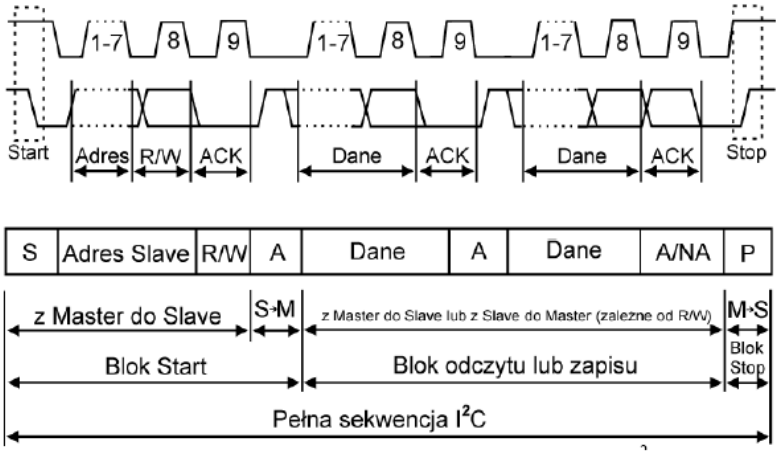
I²C do transmisji wykorzystuje dwie dwukierunkowe linie: **SDA** (linia danych, ang. *Serial Data Line*) i **SCL** (linia zegara, ang. *Serial Clock Line*). Obydwie linie są na stałe podciągnięte do źródła zasilania poprzez rezystory podciągające (ang. pull-up). I²C używa logiki dodatniej, a więc stan niski na magistrali odpowiada "0" logicznemu, natomiast stan wysoki "1" logicznej.



Wszystkie nadajniki są typu otwarty kolektor lub otwarty dren, a więc na liniach występuje tzw. iloczyn na drucie ("1" jest recesywna, a "0" dominujące). Pozwala to na wykrywanie kolizji. Każde urządzenie nadając "1" jednocześnie sprawdza, czy na magistrali rzeczywiście pojawił się stan wysoki. Jeżeli tak nie jest, oznacza to, iż inne urządzenie nadaje w tym samym czasie i urządzenie zaprzestaje nadawania.

Podstawowa wersja I²C zakłada istnienie tylko jednego urządzenia, które może inicjować transmisję (master), ale dzięki istnieniu mechanizmu detekcji kolizji, możliwa jest praca w trybie multi-master. Ponieważ dane nadawane są w kolejności od najstarszego bitu do najmłodszego, w przypadku jednoczesnego nadawania, urządzenie nadające adres o wyższym numerze wycofa się pierwsze, co wynika z binarnego sposobu zapisywania liczb. Występuje tu zatem arbitraż ze stałym przydziałem priorytetów, określonym przez adres urządzenia typu slave. Urządzenia o niższych adresach mają wyższy priorytet od urządzeń o adresach wyższych.

Protokół interfejsu I2C



- Zmiana na linii danych podczas transmisji może następować jedynie, gdy linia zegara znajduje się w stanie niskim. Nie dotyczy to specjalnych sytuacji: bitu startu i bitu stopu.
- Bit startu ma miejsce, gdy linia danych zmienia swój stan z "1" na "0", podczas wysokiego stanu linii zegara, co ma miejsce w momencie rozpoczynania każdej transmisji danych.
- Po zakończeniu transmisji generowany jest bit stopu, czyli przejście linii danych w stan wysoki przy wysokim stanie linii zegara.
- Standard zakłada magistralowe połączenie urządzeń.

Interfejs I2C

I²C stosuje się w przypadkach, gdy prostota i niski koszt są ważniejsze od wysokich prędkości transmisji. Znalazło ono zastosowanie m.in. w:

- Komunikacji pomiędzy układami scalonymi w telewizorach i innym sprzęcie RTV (jest to pierwotne miejsce zastosowania magistrali I²C)
- Odczytywaniu zegarów czasu rzeczywistego (RTC) w komputerach i urządzeniach wbudowanych
- Komunikacji z wolnymi przetwornikami cyfrowo-analogowymi i analogowo-cyfrowymi
- Komunikacji z pamięciami EEPROM, akcelerometrami, żyroskopami, magnetometrami, (czujnikami MEMS)
- Odczycie czujników diagnostycznych w komputerze (prędkość obrotu wentylatorów, temperatury procesora i ważniejszych układów na płycie głównej)
- Robotyce (czujniki przyspieszenia i odległości)
- Komunikacja z czujnikami i elementami wykonawczymi w małych systemach wbudowanych
- Dostępie do pamięci NVRAM komputera
- Sterowanie diodami LED w urządzeniach przenośnych (np. komórkach)

- Wzorując się na interfejsie I2C zostały opracowane podobne interfejsy, np. **SMBus, ACCESS.bus, VESA Display Data Channel (DDC) interface, PMBus**