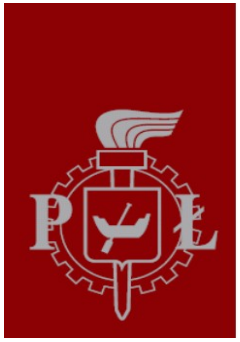


Politechnika Łódzka
Instytut Elektroniki

Programowanie Mikrokontrolerów

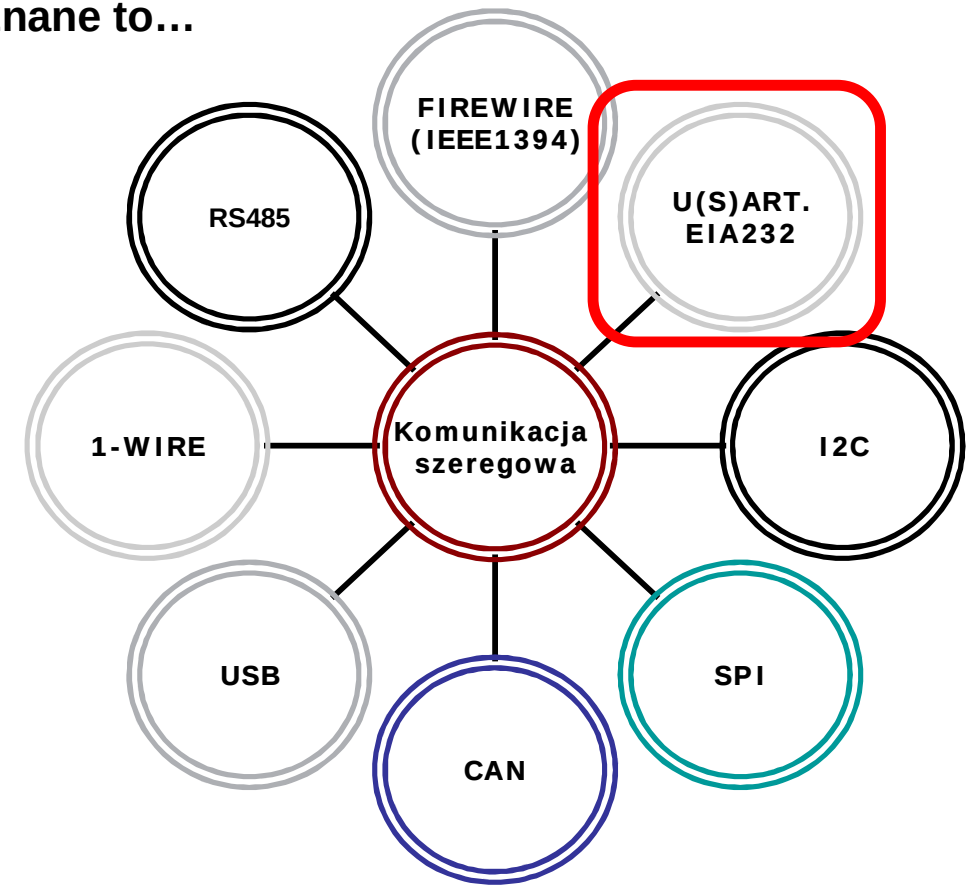
Komunikacja szeregowa w standardzie EIA232 z wykorzystaniem modułu USART.

mgr inż. Paweł Poryzała
Zakład Elektroniki Medycznej

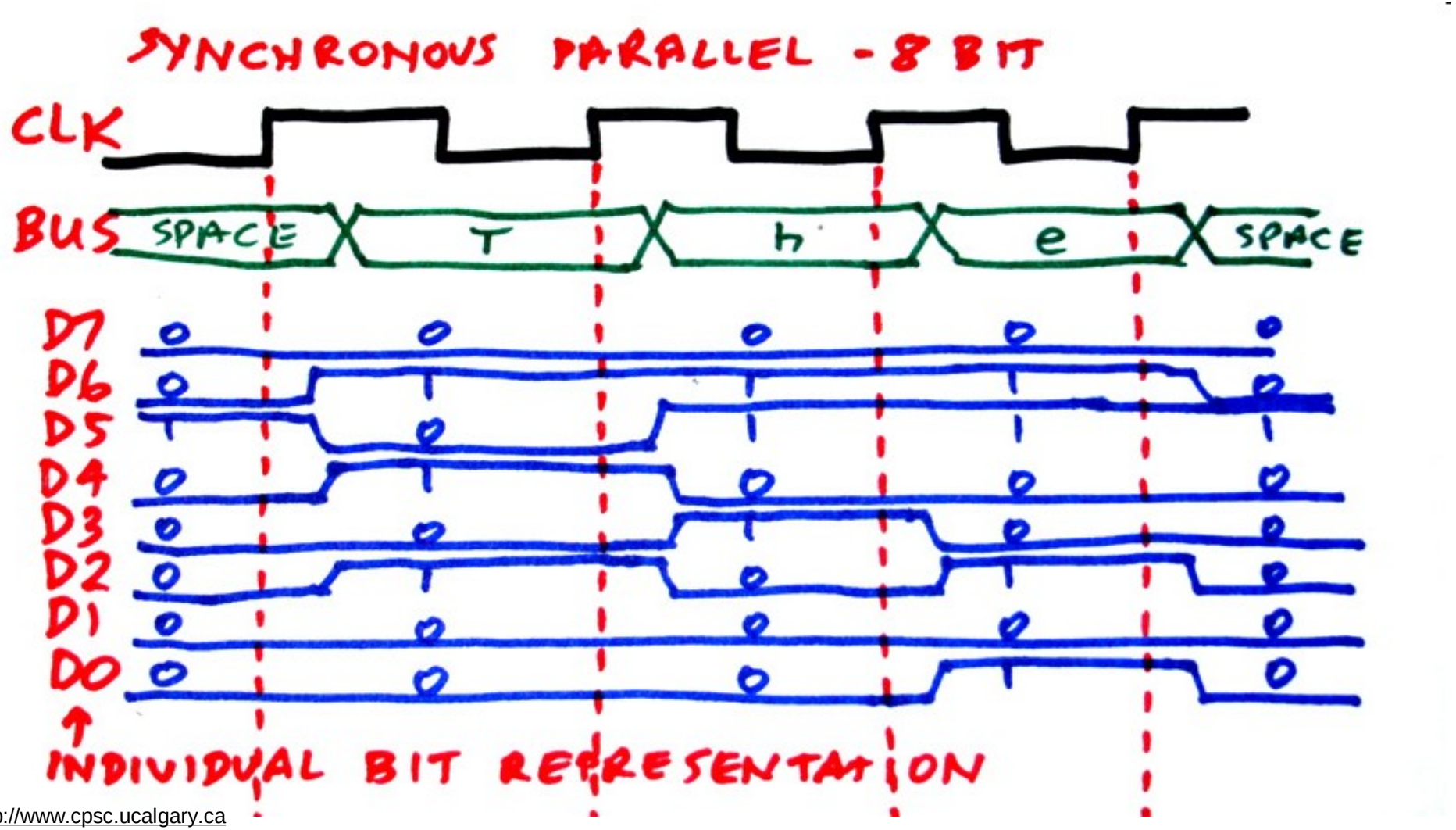
Komunikacja szeregowa

- Jakie znamy typy komunikacji szeregowej?

Prawdopodobnie najbardziej znane to…

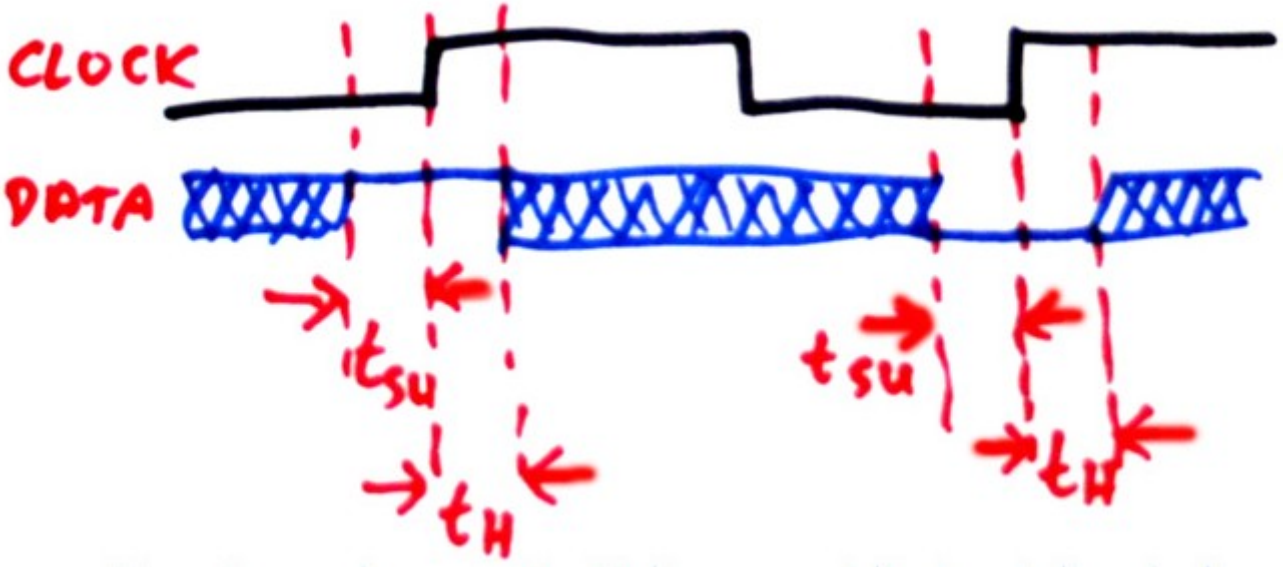


Kom. równoległa, synchroniczna



http://www.cpsc.ucalgary.ca

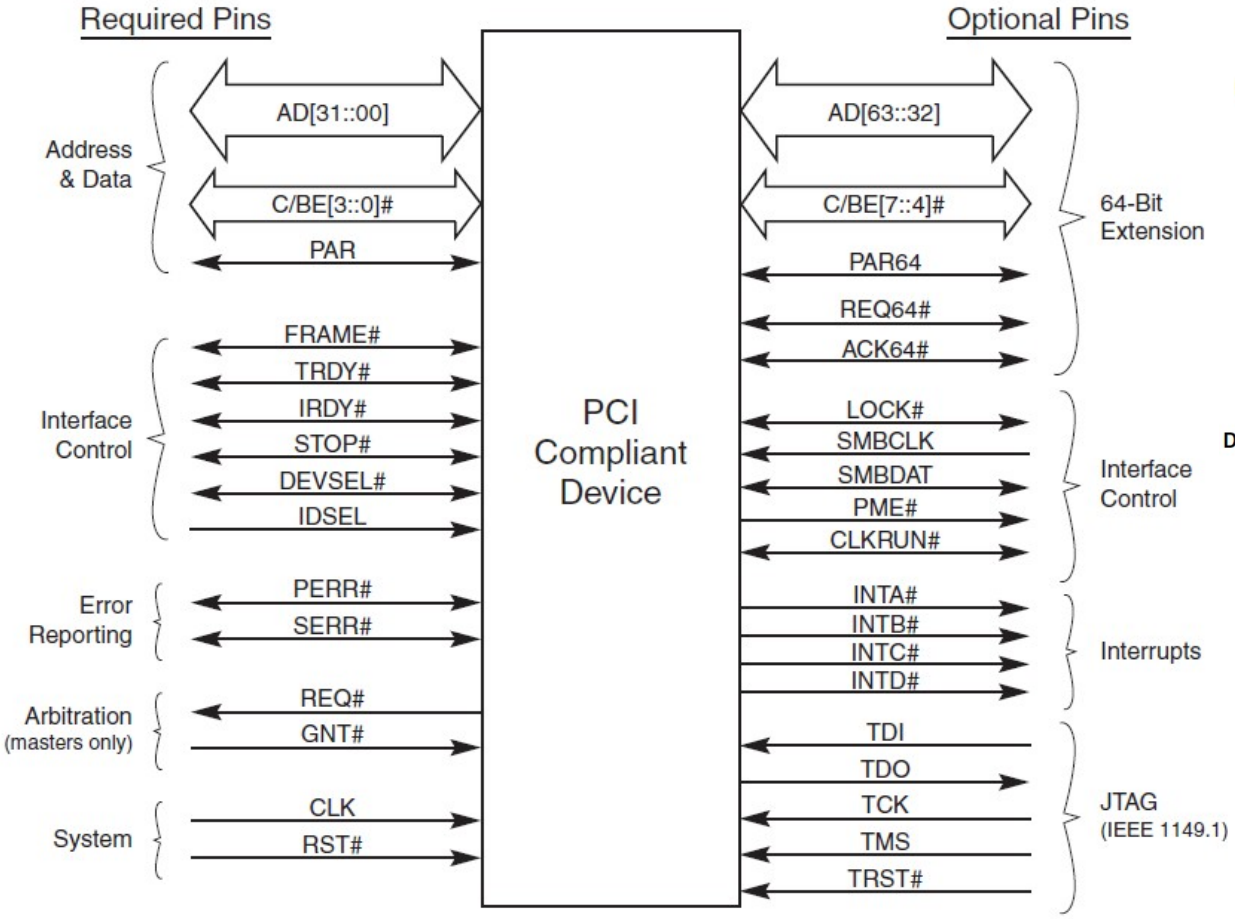
Kom. równoległa, synchroniczna

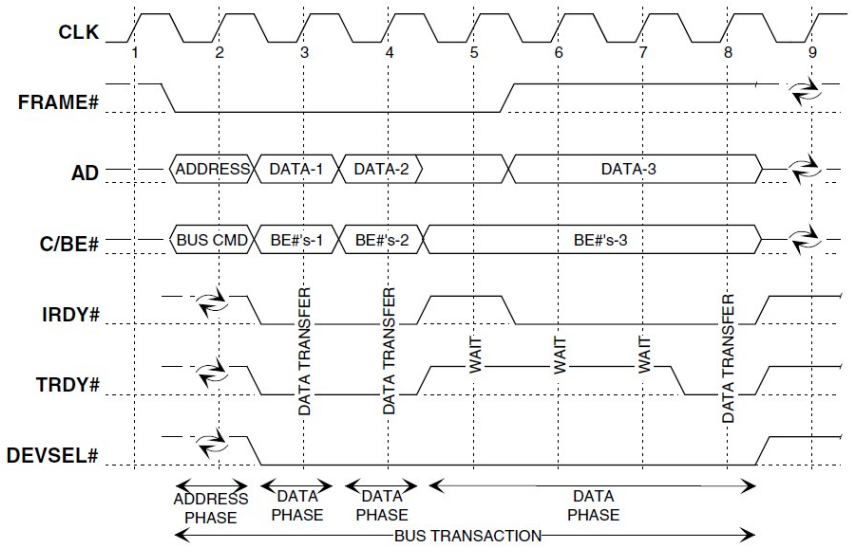


- czasy *set up* oraz *hold* w relacji do zbocza zegara synchronizującego transmisję

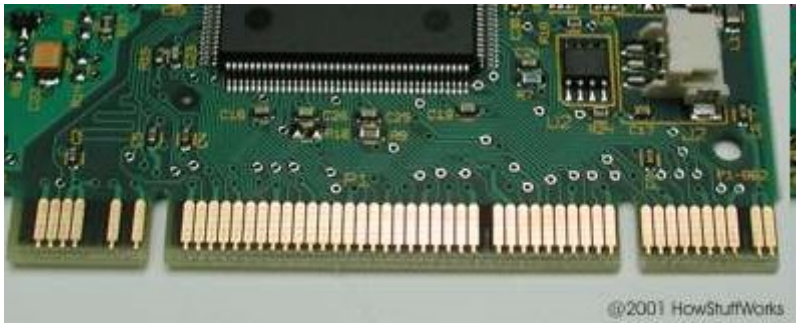
Kom. równoległa, synchroniczna

- przykłady: PCI (v2.3)

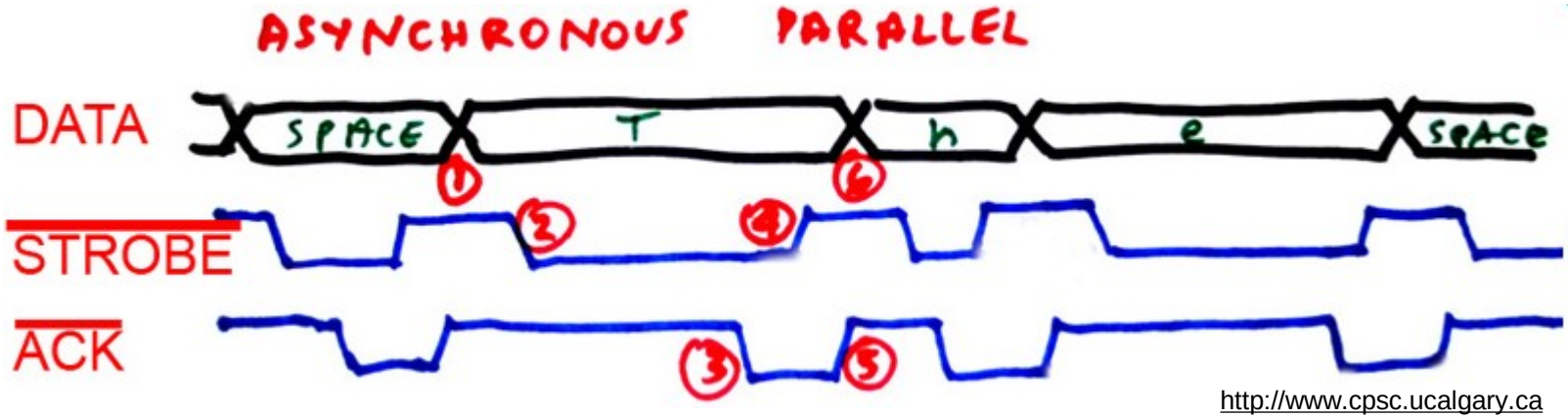




uniwersalna, 32-bitowa karta PCI (3.3V & 5V)



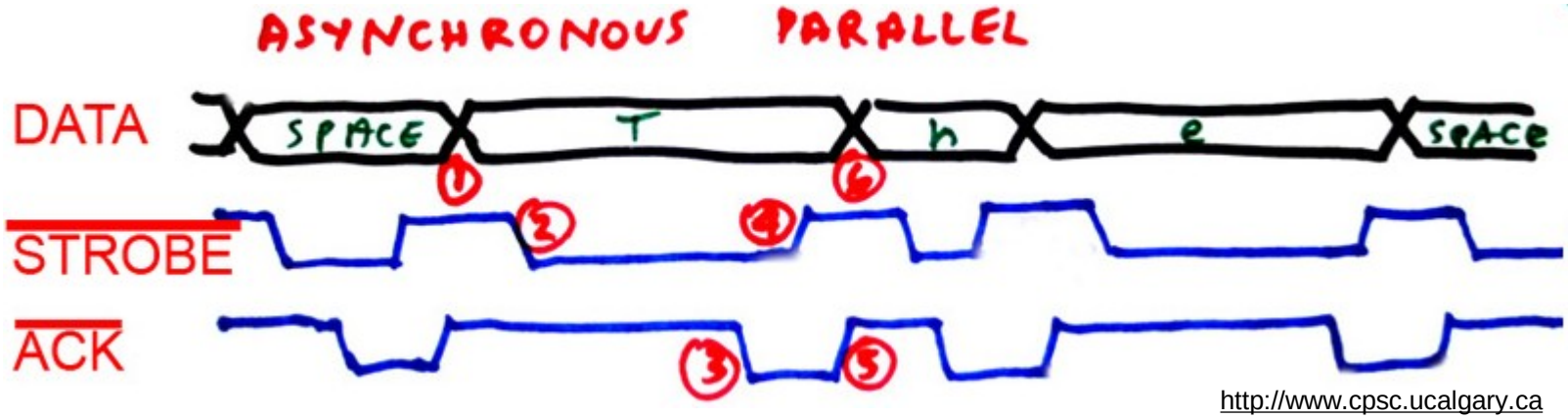
Komunikacja równoległa, asynchroniczna



http://www.cpsc.ucalgary.ca

- problem z utrzymaniem pełnej synchronizacji na dłuższej magistrali,
- metoda *handshake* za pomocą dodatkowych sygnałów (tutaj nSTROBE oraz nACK),
- przykład: port równoległy LPT (komunikacja z drukarką w trybie SPP - *Standard Parallel Port*), SCSI-1 (1986 rok)

Komunikacja równoległa, asynchroniczna



http://www.cpsc.ucalgary.ca

(1) litera „T” jest wystawiona na magistrali danych, co sygnalizowane jest zboczem opadającym na linii nSTROBE (2)

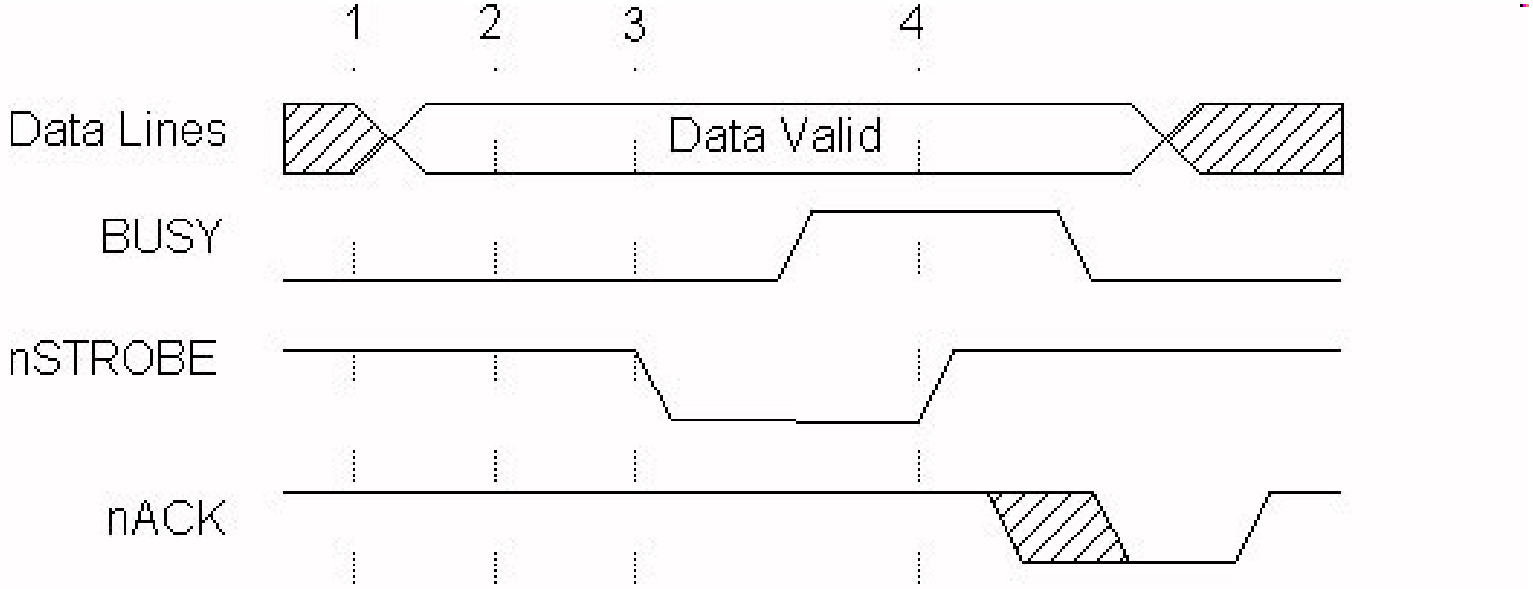
(3) drukarka odbiera dane z magistrali, co sygnalizowane jest zboczem opadającym na linii nACK

(4) z magistrali zdjęty jest sygnał nSTROBE

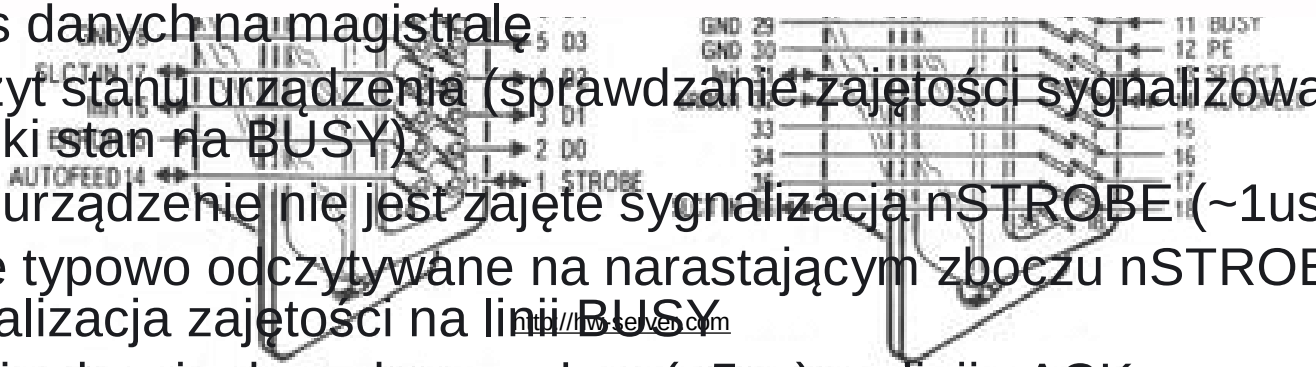
(5) drukarka zdejmuje sygnał nACK (na port nie zostaną wystawione nowe dane dopóki nACK jest w stanie niskim)

(6) transmisja kolejnego znaku

Komunikacja równoległa, asynchroniczna – protokół Centronics oraz SPP



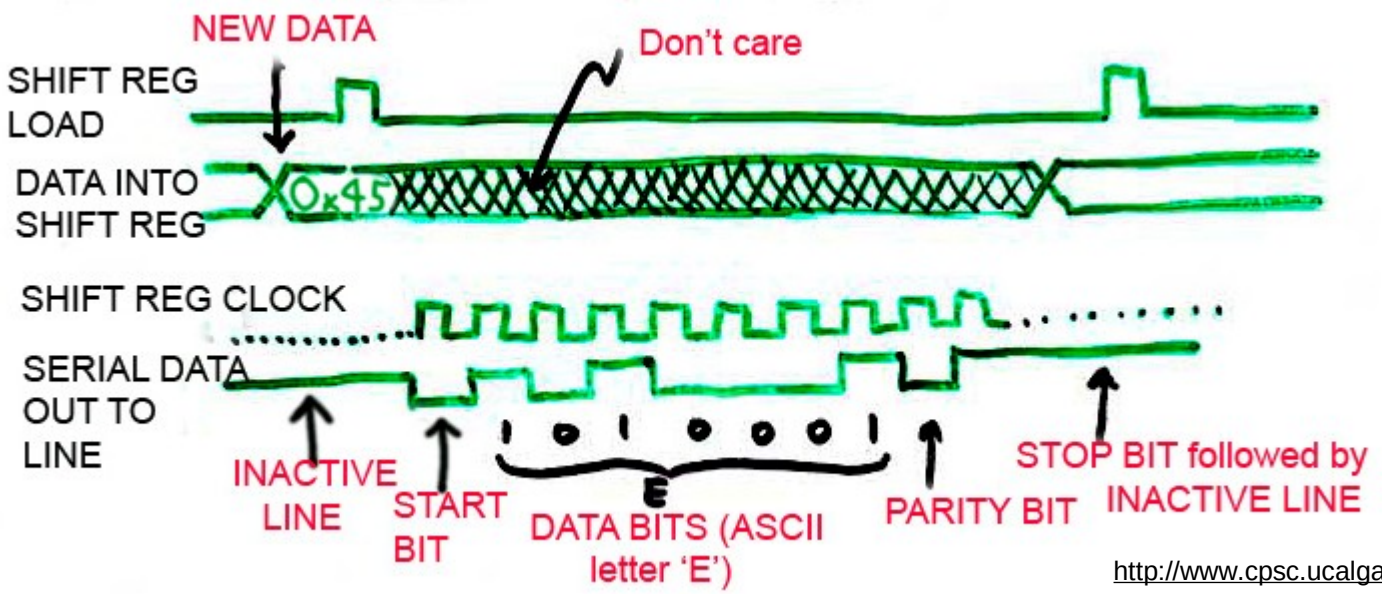
(1) Zapis danych na magistralę
(2) Odczyt stanu urządzenia (sprawdzanie zajętości sygnalizowanej przez wysoki stan na BUSY)
(3) Jeśli urządzenie nie jest zajęte sygnalizacja nSTROBE (~1us)
(4) Dane typowo odczytywane na narastającym zboczu nSTROBE oraz sygnalizacja zajętości na linii BUSY
(5) Potwierdzenie danych sygnałem (~5us) na linii nACK



Komunikacja szeregowa

- wysyłanie danych sekwencyjnie, kolejno po jednym bicie danych w konkretnej chwili czasu w kanale transmisji,
- główne cechy:
  - mniejszy koszt przewodów lub innego medium transmisji, zajmują mniej miejsca (przykład kable ATA oraz SATA),
  - brak problemu synchronizacji danych przesyłanych równolegle (*clock skew*), szczególnie w przypadku transmisji asynchronicznej,
  - mniejsze przesłuchy pomiędzy liniami transmisji.
- przykłady: RS232, RS422, RS485, I2C, USB, FireWire, Ethernet, FibreCable, MIDI, DMX512, SATA, PIC Express, ModBus

Komunikacja szeregowa, asynchroniczna



http://www.cpsc.ucalgary.ca

- urządzenie nadawcze oraz odbiorcze muszą pracować z tą samą prędkością (różnica do max. 5%)

Komunikacja szeregowa, asynchroniczna

- zegar: 14,7456 MHz, tolerancja ±100ppm, baud rate: 115 200 bps
- UBRR = 7 wyznaczone dla idealnego przypadku
- zmiana zegara po stronie nadawczej o 100ppm (0,01%) da 14,7441 MHz
- faktyczny baud rate: 115 188bps (zmiana o ułamek procenta)

$$BAUD = \frac{f_{OSC}}{16(\boldsymbol{UBRR}+1)}$$

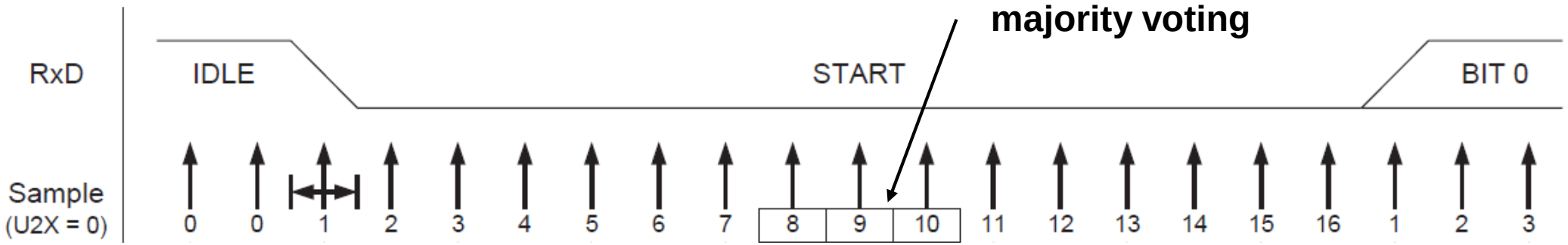
Komunikacja szeregowa, asynchroniczna

- zegar: 8,0000 MHz, tolerancja ±100ppm, baud rate: 115 200 bps
- wyznaczony UBRR = 3,34, konfiguracja UBRR = 3
- faktyczny baud rate: 125 000bps (różnica 8,5%)

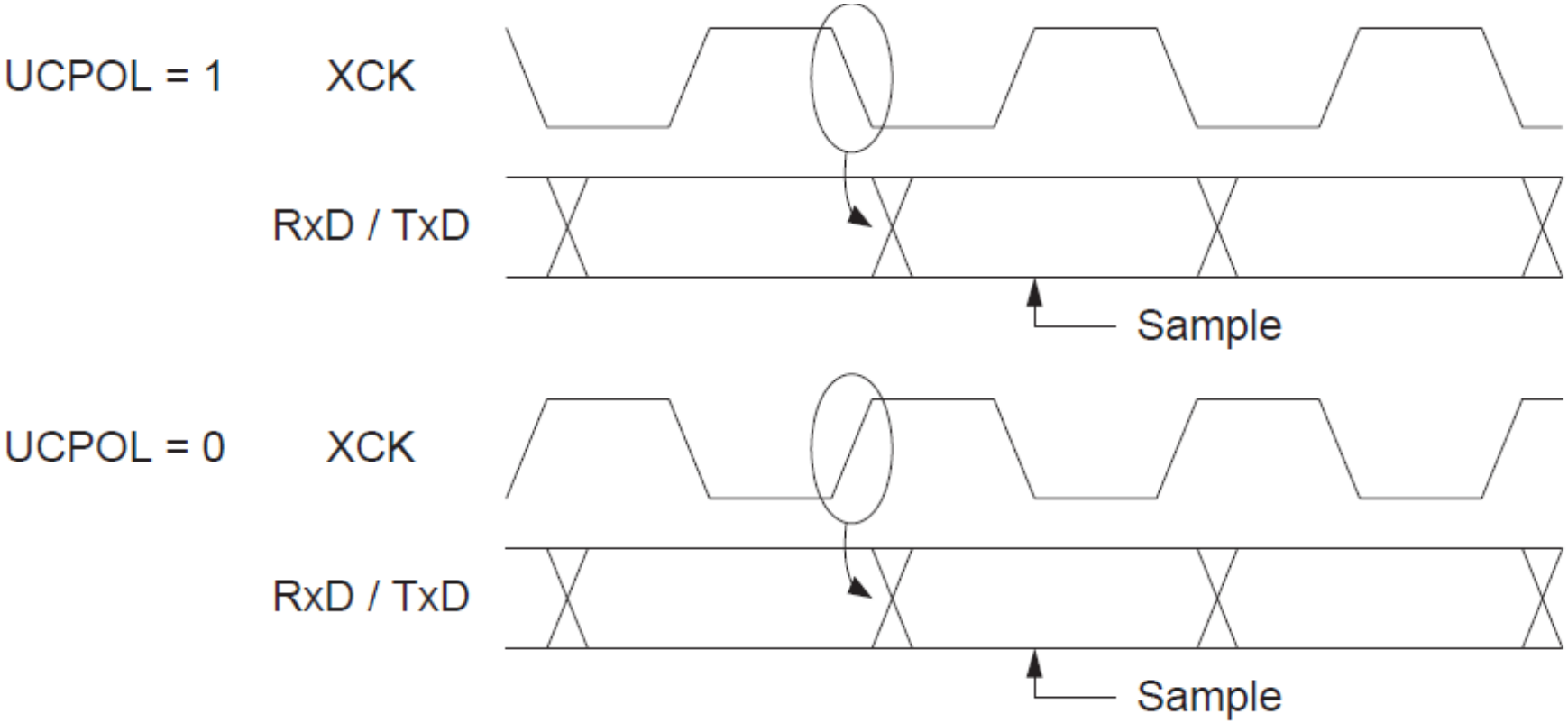
$$BAUD = \frac{f_{OSC}}{16(UBRR + 1)}$$

Komunikacja szeregowa, asynchroniczna

- zwykle 16-krotne nadpróbkowanie w celu odnalezienia środka bitu, gdy to się powiedzie kolejny bit próbkowany jest za 16 cykli
- przy 16-krotnym nadpróbkowaniu dopuszczalna jest różnica Baud Rates na poziomie 5% (przy założeniu ramki 8N1)



Komunikacja szeregowa, synchroniczna



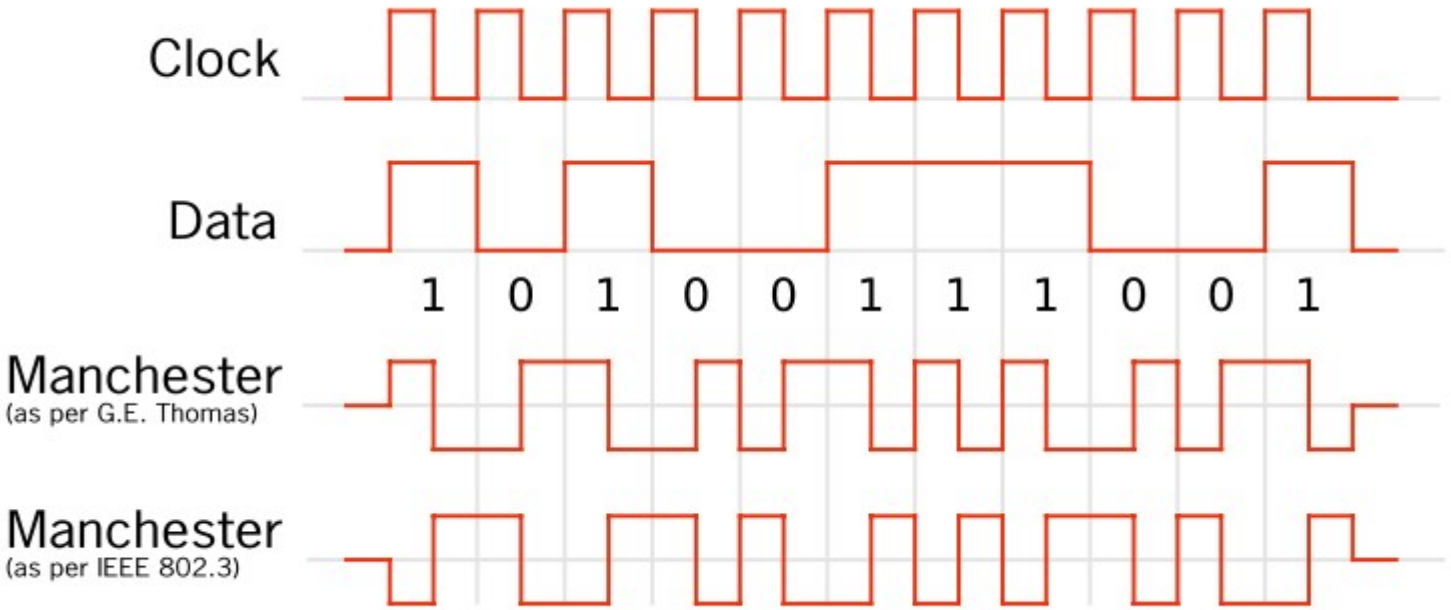
- np.: I2C, SPI, tryb synchronicznej pracy interfejsu USART

Self-clocking line code

- kodowanie 8B/10B używane w FireWire, PCI Express, Serial ATA, DVI oraz HDMI, USB 3.0
  - line code that maps 8-bit symbols to 10-bit symbols to achieve DC-balance and bounded disparity, and yet provide enough state changes to allow reasonable clock recovery. This means that the difference between the count of 1s and 0s in a string of at least 20 bits is no more than 2, and that there are not more than five 1s or 0s in a row. This helps to reduce the demand for the lower bandwidth limit of the channel necessary to transfer the signal.

Self-clocking line code

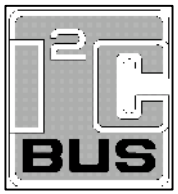
- kodowanie Manchester używane w sieciach Ethernet



Komunikacja szeregowa - podział

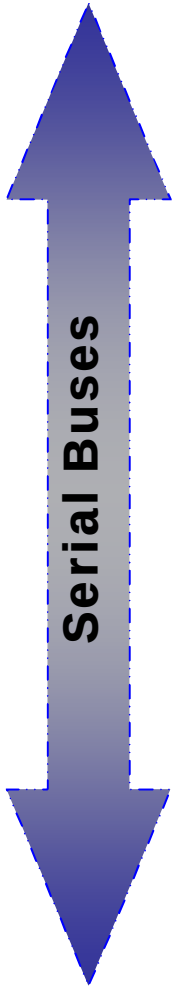
SPI

UART



- Typowo podział na:
  - Prędkość, szybkość przesyłania danych,
  - Liczbę możliwych do podłączenia na jednej magistrali urządzeń,
  - Możliwą długość przewodów.
- UWAGA:
  - Wszystko określane jest dla założeń danego standardu oraz konkretnego zastosowania (np. obniżając prędkość transmisji, godząc się na inne ograniczenia można np. wydłużyć połączenia).

Komunikacja szeregowa – UART:



- UART – Universal Asynchronous Receiver Transmitter:
  - Standard rozwinięty już w latach 60’,
  - Prosty, uniwersalny, dobrze udokumentowany,
  - Wolna komunikacja: max. 1Mbit/s,
  - Maksymalnie dwa urządzenia,
  - Minimum komunikacji: po jednym przewodzie komunikacyjnym w każdym z kierunków plus wspólna masa sygnałowa,
  - Asynchroniczna – odbiornik i nadajnik muszą wcześniej znać szybkość transmisji (ew. U**S**ART),
  - Bity START i STOP normują przesyłanie,
  - Możliwe dołączenie informacji o parzystości (b. prosta kontrola błędów).

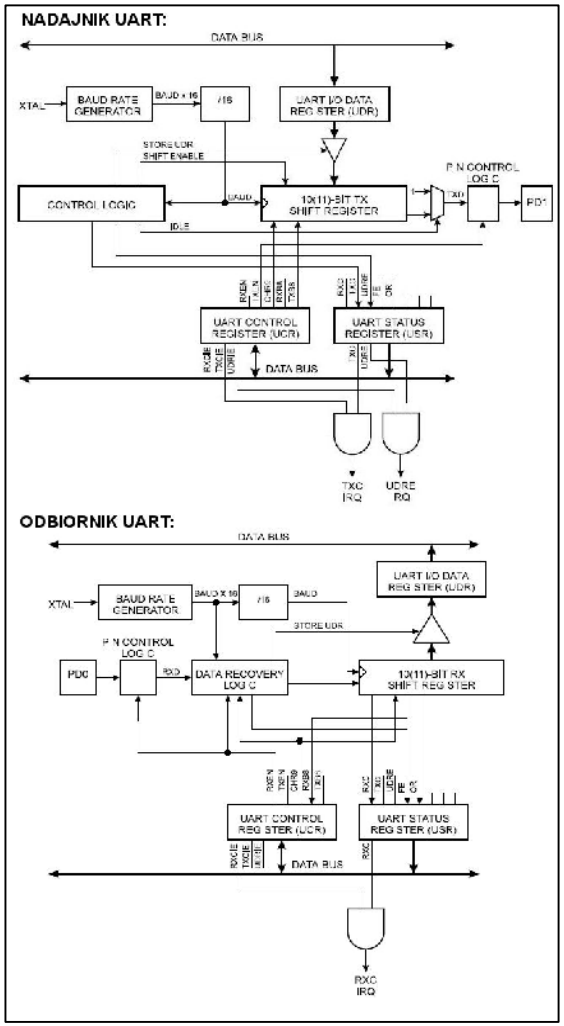
Komunikacja szeregowa

- Dwa podstawowe typy komunikacji szeregowej:
  - synchroniczna – równolegle z ciągiem bitów przesyłany jest sygnał synchronizujący, określający kiedy dane są ważne (*valid*),
  - asynchroniczny – dane nie są związane z żadnym sygnałem synchronizującym – parametry transmisji trzeba ustawić wcześniej – ręcznie, a do danych należy dodać informacje pomocne w synchronizacji danych wysyłanych i odbieranych (np.: ramka *8, N, 1*; *8, E, 2*),

- Nazewnictwo:
  - Port full-dupleksowy (dwukierunkowy) – może równocześnie nadawać i odbierać dane (analogia: telefon),
  - Port half-dupleksowy (jednokierunkowy) – jednoczesny odbiór i nadawanie nie jest możliwe (analogia: CB radio).

- W AVR'ach mamy w pełni **dupleksowy** (dwukierunkowy, osobne rejestry nadawania i odbioru) **USART** – do wyboru transmisja asynchroniczna lub synchroniczna.

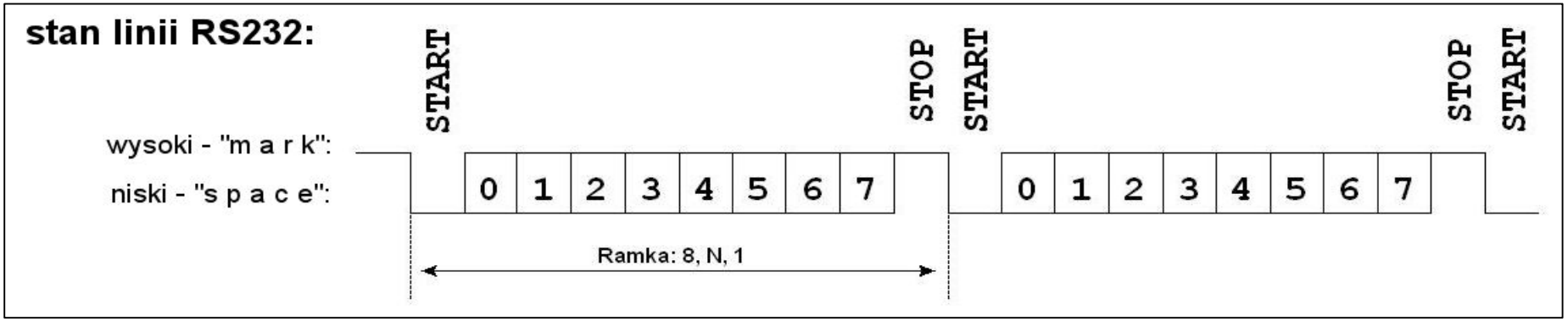
Informacje: układ USART



- **USART** – *Universal Synchronous Asynchronous Receiver/Transmitter* - sprzętowy kontroler transmisji szeregowej, który zamienia dane równoległe (wpisywane do rejestru) na postać szeregową, która wysyłana jest do odbiornika, np. poprzez EIA232 (po dopasowaniu poziomów logicznych).

- Moduł USART wykonuje wszelkie zadania związane z transmisją danych: ich synchronizację, sprawdzanie parzystości (jeśli jest wykorzystywane), poprawność ramek danych i inne.

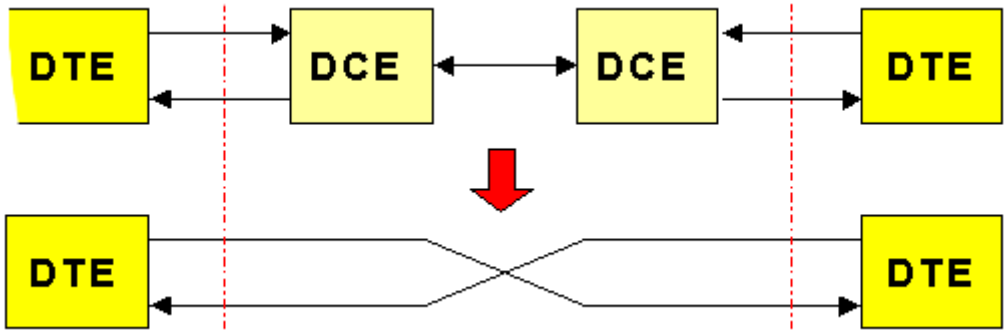
EIA232 – specyfikacja



- Brak sygnału synchronizującego upraszcza znacznie transmisję, ale koniecznie należy pamiętać, że:
  - bity wysyłane są z daną częstotliwością (*baud rate*) i ta musi być jednakowa dla odbiornika i nadajnika,
  - odbiornik może zacząć odbiór w nieodpowiednim momencie, (rozsynchronizowanie) - na ponowną synchronizację potrzeba czasu,
  - potrzebne są dodatkowe bity aby właściwie zaznaczyć początek i koniec właściwych danych.
- W tym celu protokół określa następujące bity dodawane do transmitowanych danych: **bit startu,** bit parzystości**, bit stopu**

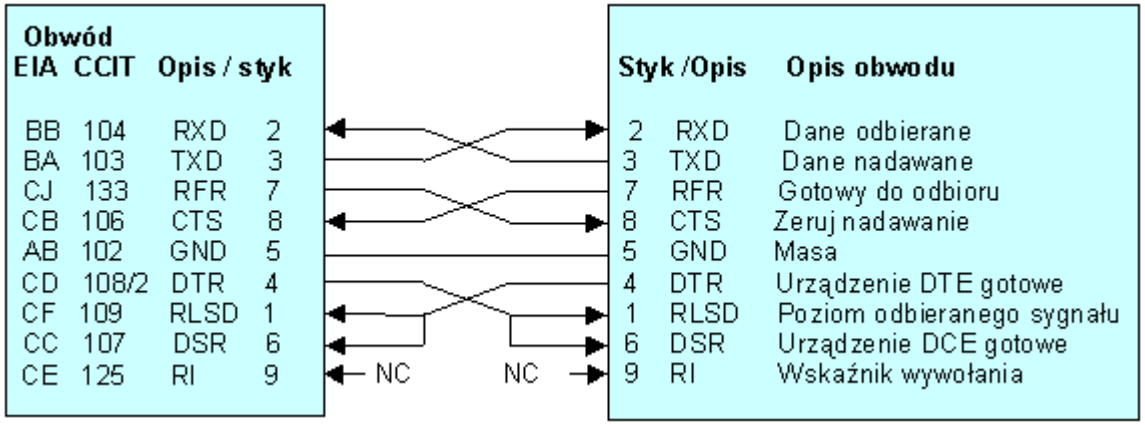
EIA232 – specyfikacja

- DTE: *Data Terminal Equipment* - urządzenie do przetwarzania danych
- DCE: *Data Communication Equipment* – urządzenie do transmisji danych



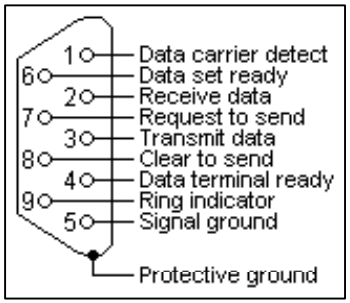
„tradycyjne” połączenie dwu DTE za pośrednictwem DCE

bezpośrednie połączenie dwu DTE (kabel Null-Modem)



Sposób połączenia dwu urządzeń oraz opis styków interfejsów wg norm EIA oraz CCIT

EIA232 – specyfikacja

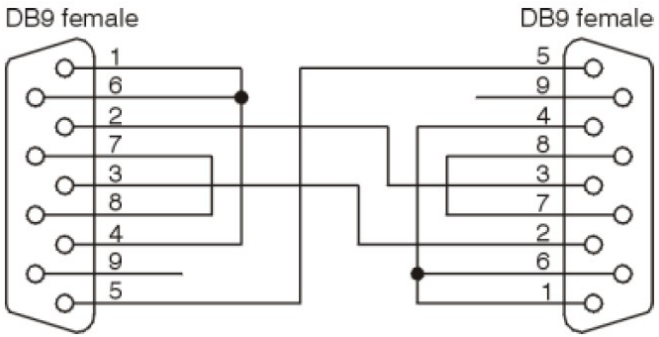


- Rozkład wyprowadzeń gniazda DB9
  - DTE: *Data Terminal Equipment*
  - DCE: *Data Communication Equipment*

oznacz.:	nazwa	kierunek	opis	wypr.:
TXD	transmitted data	DCE <– DTE	dane nadawane z DTE do DCE, przy braku transmisji stan MARK	3(M), 2(F)
RXD	received data	DCE –> DTE	dane odbierane przez DTE z DCE, przy braku transmisji stan MARK	2(M), 3(F)
RTS	request to send	DCE <– DTE	żądanie nadawania, sygnalizuje gotowość DTE	7
CTS	clear to send	DCE –> DTE	gotowość do odbioru przez DCE	8
DSR	DCE ready	DCE –> DTE	gotowość DCE do prowadzenia transmisji	6
DTR	DTE ready	DCE <– DTE	gotowość DTE do prowadzenia transmisji	4
DCD	carrier detect	DCE –> DTE	wykrycie nośnej, wykorzystywane przy modemach	1
GND	signal ground	DCE – DTE	masa sygnałowa, poziom odniesienia dla wszystkich sygnałów	5
RI	ring indicator	DCE –> DTE	sygnał dzwonienia, gdy DTE jest modemem	9

EIA232 – połączenie

Połączenie NULL MODEM:

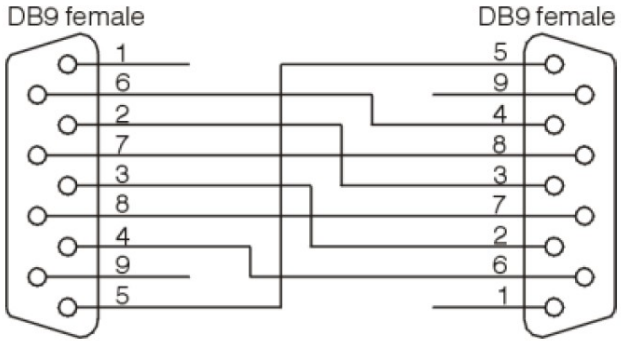


Złącze 1	Złącze 2	Funkcja
2	3	Rx ← Tx
3	2	Tx → Rx
5	5	Signal ground
1 + 4 + 6	-	DTR → CD + DTR
-	1 + 4 + 6	DTR → CD + DTR
7 + 8	-	RTS → CTS
-	7 + 8	RTS → CTS

- najłatwiejszy sposób na połączenie dwu urządzeń przez EIA232 – do transmisji wykorzystywane są jedynie linie
  - Tx (3),
  - Rx (2),
  - GND (5).
- zwarte sygnały kontroli transmisji – zakładamy, że zarówno odbiornik, jak i nadajnik są zawsze gotowe do wymiany danych.

EIA232 – połączenie

Połączenie z pełnym potwierdzaniem transmisji:

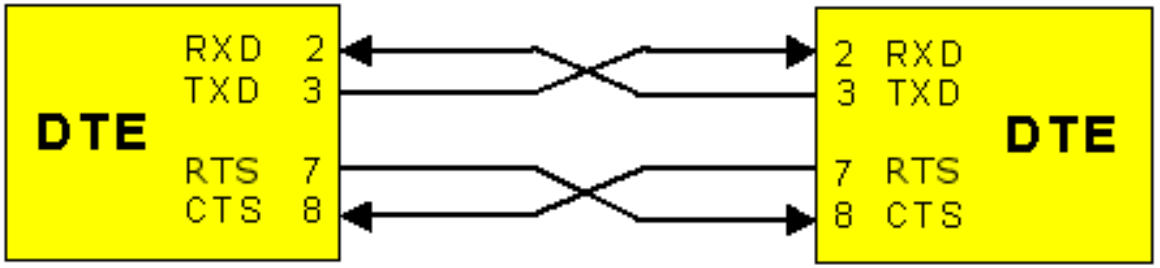


Złącze 1	Złącze 2	Funkcja
2	3	Rx ← Tx
3	2	Tx → Rx
4	6	DTR → DSR
5	5	Signal ground
6	4	DSR ← DTR
7	8	RTS → CTS
8	7	CTS ← RTS

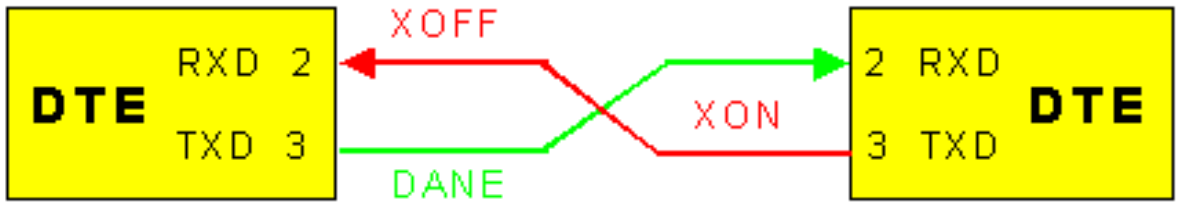
- Pełna kontrola nad transmisją danych.
- Dodatkowe warianty połączenia:
  - Połączenie z częściowym potwierdzeniem
  - Połączenie z potwierdzeniem zwrotnym

EIA232 – sterowanie przepływem

- Sterowanie sprzętowe, z wykorzystaniem linii RTS oraz CTS



- Sterowanie programowe, z wykorzystanie znaków sterujących ASCII XON oraz XOFF



XON: transmitter ON, XOFF: transmitter OFF

EIA232 – poziomy logiczne

RS232 voltage values

Level	Transmitter capable (V)	Receiver capable (V)
Logical 0 space state	+5 ... +15	+3 ... +25
Logical 1 mark state	-5 ... -15	-3 ... -25
Undefined	-	-3 ... +3

RS232 cable length according to Texas Instruments

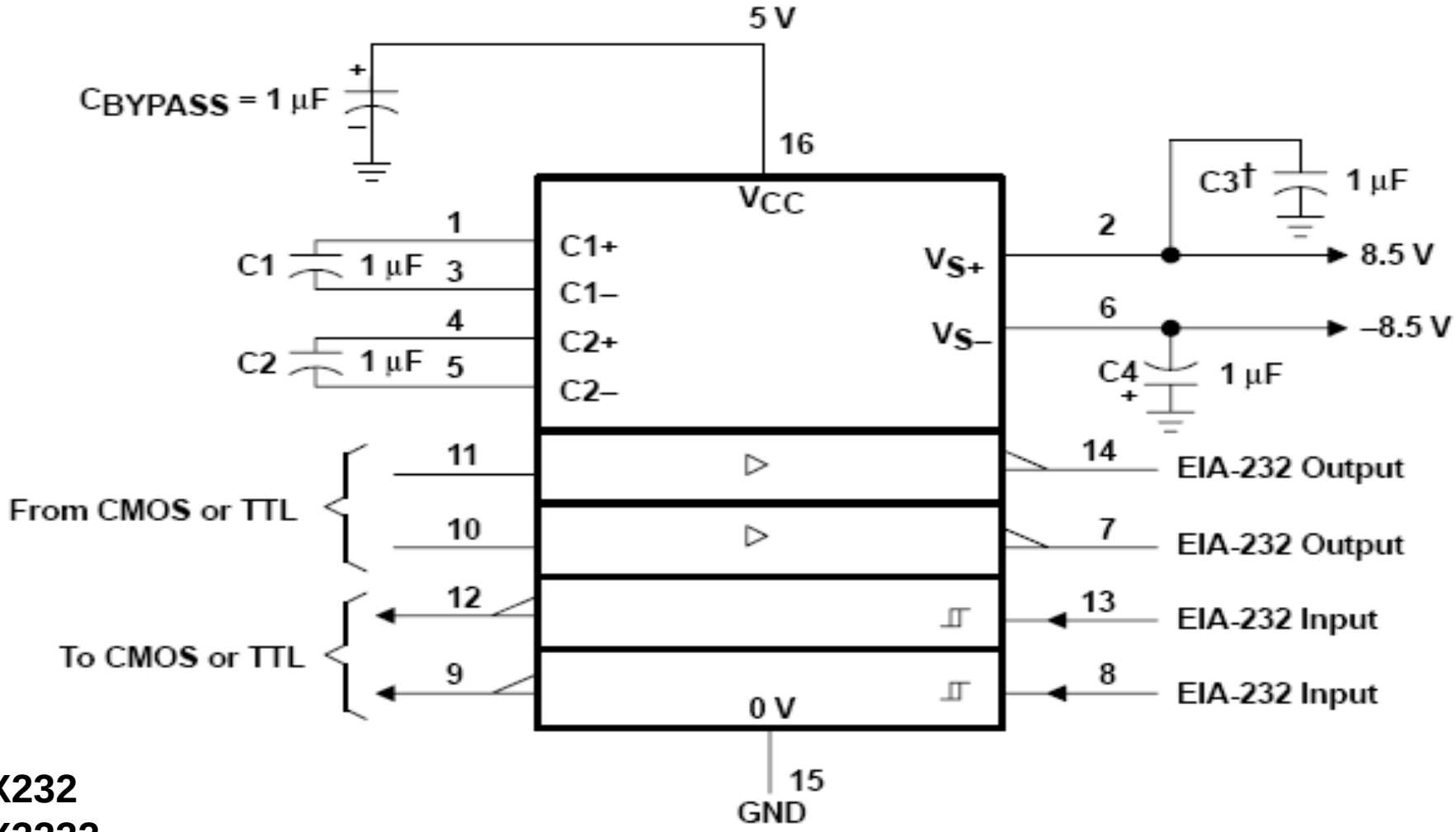
Baud rate	Maximum cable length (ft)
19200	50
9600	500
4800	1000
2400	3000

- Napięcia na liniach:
  - Stan wysoki (*mark state*) - napięcie ujemne,
  - Stan niski (*space state*) - napięcie dodatnie

Napięcia te mają wpływ na maksymalną długość kabla, szybkość transmisji a przede wszystkim na mogące wystąpić błędy (ew. można stosować RS485).

- Długość linii RS232:
  - Standard określa jednoznacznie: 50ft.
  - Zmniejszając *baud rate* można jednak kabel wydłużyć – wg tabeli.

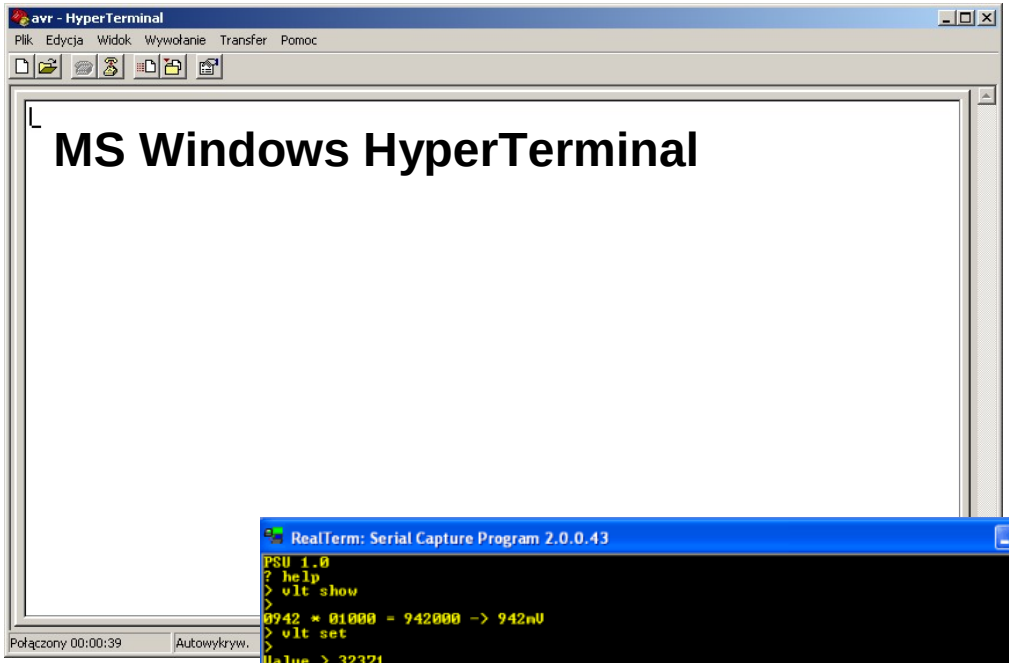
EIA232 – dopasowanie poziomów logicznych

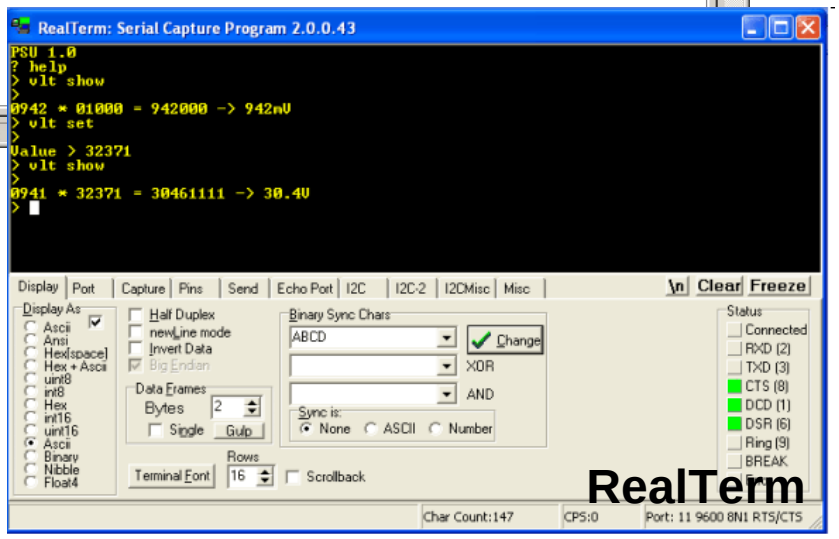


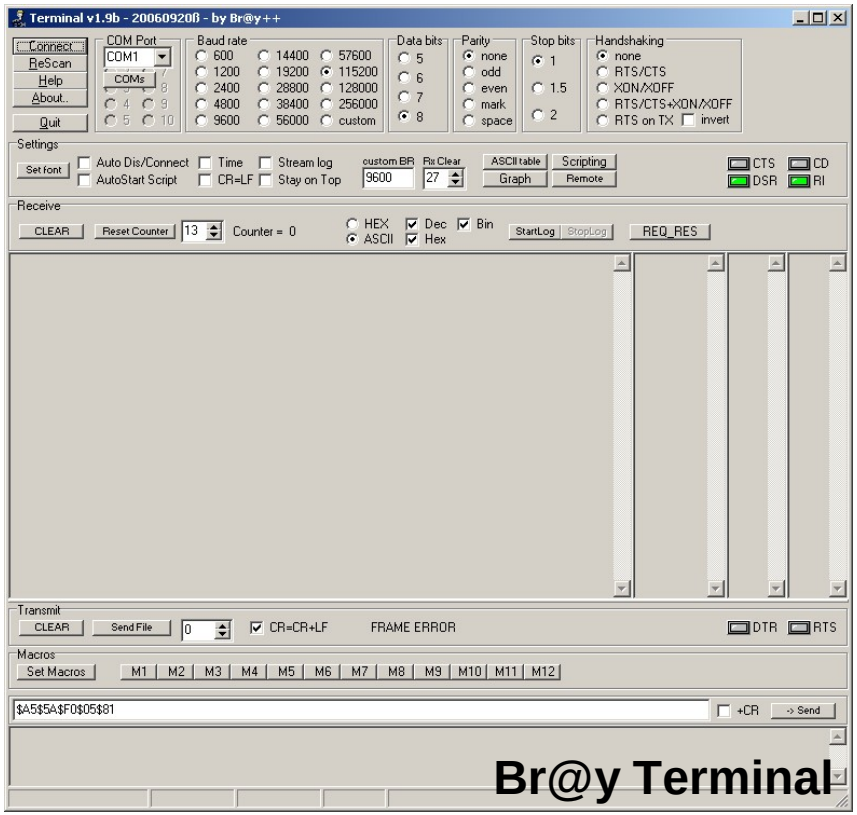
MAX232
MAX3232

źródło: TI MAX232 Datasheet

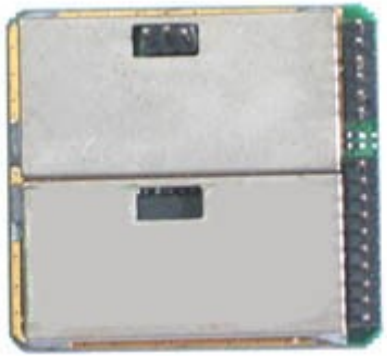
Oprogramowanie terminalowe - PC







Zastosowania



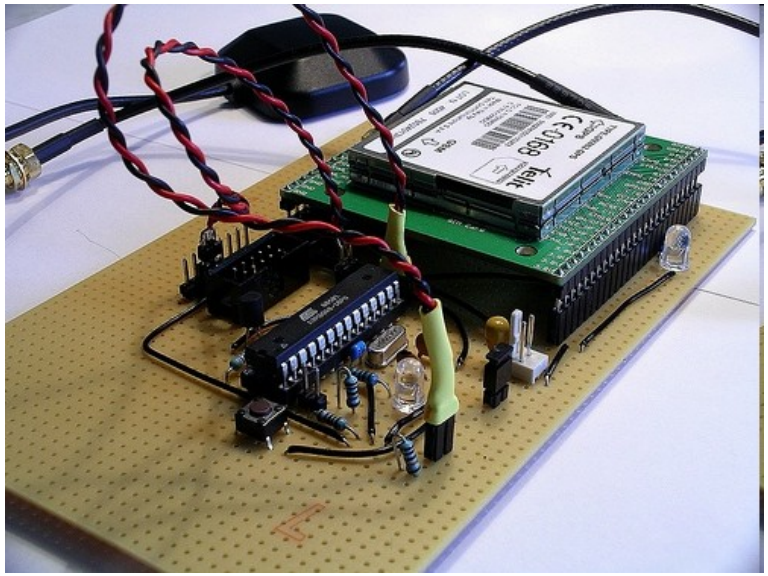
GPS

PC (serial port)

PDA

Wireless RC232 Modules

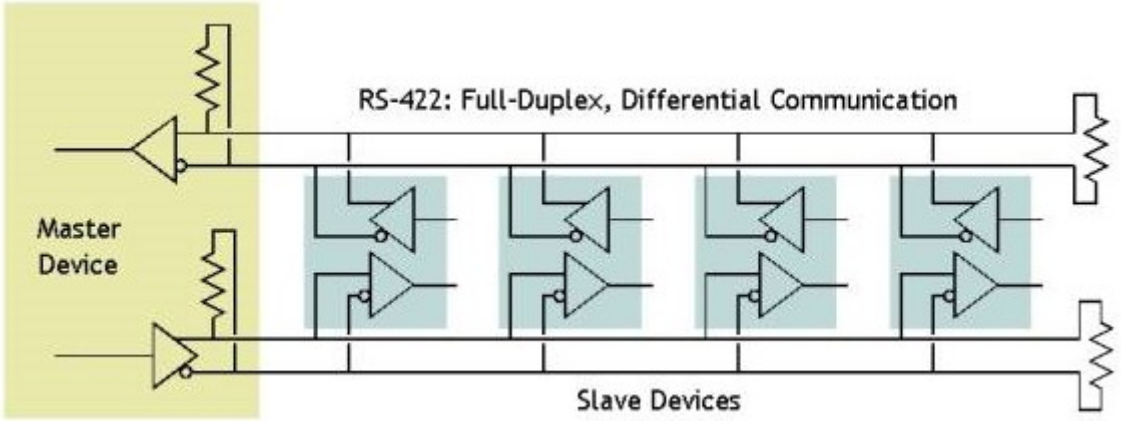
GSM

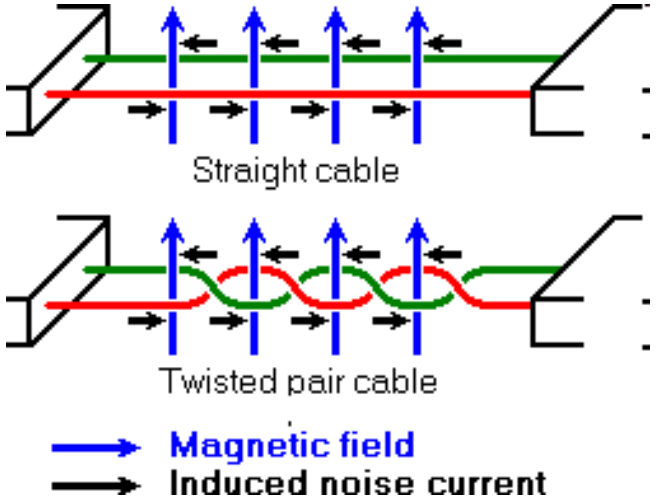


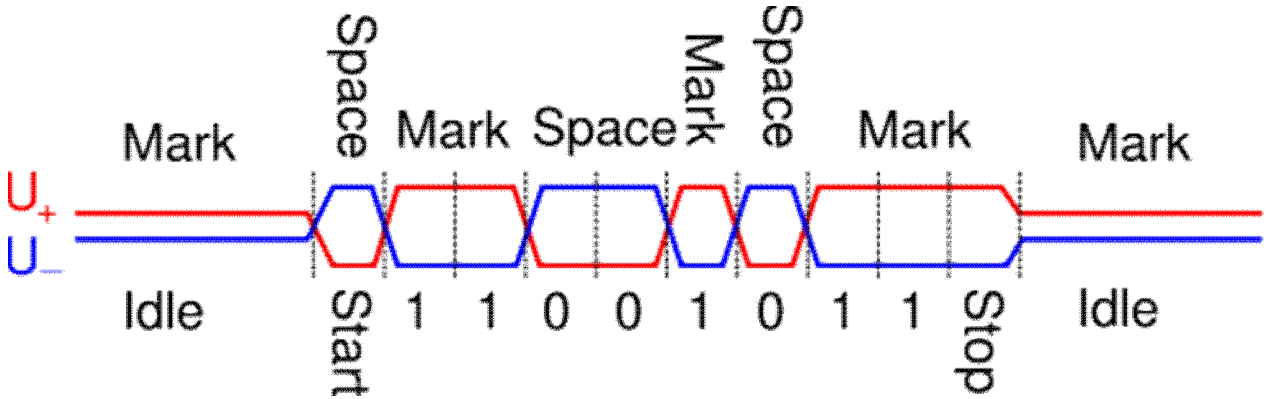
Bootloader'y
Interfejsy debugowania
I inne…

Powiązane standardy – EIA422

- a high-speed system similar to RS-232 but with differential signaling

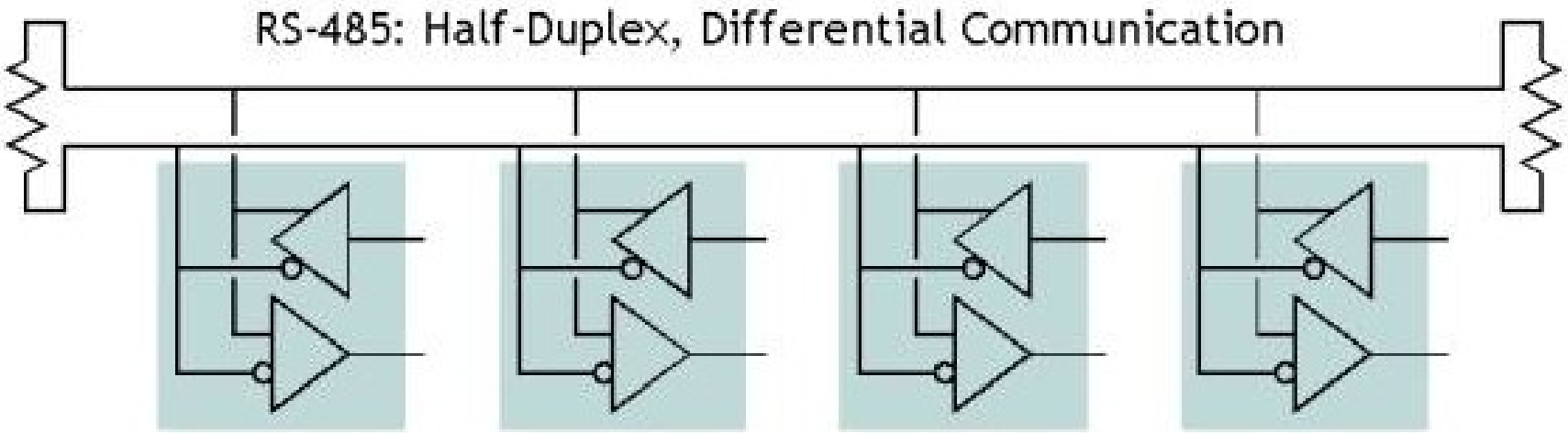






Powiązane standardy – EIA485

- EIA-485 can be used as a bus in multidrop configurations



Powiązane standardy - porównanie

	RS232	RS423	RS422	RS485
Differential	no	no	yes	yes
Max number of drivers Max number of receivers	1 1	1 10	1 10	32 32
Modes of operation	half duplex full duplex	half duplex	half duplex	half duplex
Network topology	point-to-point	multidrop	multidrop	multipoint
Max distance (acc. standard)	15 m	1200 m	1200 m	1200 m
Max speed at 12 m Max speed at 1200 m	20 kbs (1 kbs)	100 kbs 1 kbs	10 Mbs 100 kbs	35 Mbs 100 kbs
Max slew rate	30 V/µs	adjustable	n/a	n/a
Receiver input resistance	3..7 kΩ	≧ 4 kΩ	≧ 4 kΩ	≧ 12 kΩ
Driver load impedance	3..7 kΩ	≧ 450 Ω	100 Ω	54 Ω
Receiver input sensitivity	±3 V	±200 mV	±200 mV	±200 mV
Receiver input range	±15 V	±12 V	±10 V	-7..12 V
Max driver output voltage	±25 V	±6 V	±6 V	-7..12 V
Min driver output voltage (with load)	±5 V	±3.6 V	±2.0 V	±1.5 V

Kod przykładowy: Inicjalizacja

```
// !-- 1 --!
#define FOSC 14745600   // Clock Speed
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1

void main( void ) {
    ...
    USART_Init ( MYUBRR );
    ...
}

void USART_Init( unsigned int ubrr) {

    // !-- 2 --!
    UBRRH = (unsigned char)(ubrr>>8);
    UBRRL = (unsigned char)ubrr;

    // !-- 3 --!
    UCSRB = (1 << RXEN)|(1 << TXEN);

    // !-- 4 --!
    UCSRC = (1<<URSEL)|(3<<UCSZ0);
}
```

Komentarz do kodu:

- Definiując odpowiednie **FOSC** (podane w Hz) oraz **BAUD** unikamy szukania odpowiedniej wartości w tabelach na str. 159,

Inicjalizacja USART:

2. ustawiamy odpowiednią prędkość transmisji (przekazaną w definicji **MYUBRR** przy wywołaniu funkcji inicjalizacji),
3. włączamy odbiornik oraz nadajnik USARTu,
4. konfigurujemy parametry ramki interfejsu: 8 bitów danych, jeden bit stopu brak parzystości

   UWAGA na bit URSEL!!!

źródło: ATmega8(L) Datasheet

Kod przykładowy: Transmisja

```
void USART_Transmit
        ( unsigned char data )
{
   // !-- 1 --!
   while ( !( UCSRA&(1<<UDRE)) )
        ;

   // !-- 2 --!
   UDR = data;
}
```

Funkcja wysyłająca pojedynczy bajt:

1. Pooling bitu UDRE z rejestru UCSRA – oczekiwanie na zwolnienie bufora danych mikrokontrolera (zabezpieczenie przed zamazaniem nie wysłanych jeszcze danych),
2. Zapisanie danych do rejestru UDR – transmisja następuje automatycznie.

źródło: ATmega8(L) Datasheet

Kod przykładowy: Odbiór

```
unsigned char USART_Receive( void )
{
   // !-- 1 --!
   while ( !(UCSRA & (1<<RXC)) )
        ;

    // !-- 2 --!
   return UDR;
}
```

Funkcja odbierająca pojedynczy bajt:

1. Pooling bitu RXC z rejestru UCSRA – oczekiwanie na zakończenie odbioru danych przez blok USART
2. Odczyt odebranych danych z rejestru UDR oraz zwrócenie ich jako wartość funkcji.

źródło: ATmega8(L) Datasheet

Więcej o UART / Serial Port

- Wikipedia EN:
  - http://en.wikipedia.org/wiki/RS-232
- Beyond Logic, Interfacing The RS232 / Serial Port:
  - http://www.beyondlogic.org/serial/serial.htm
- Introduction to Serial Communications:
  - http://www.taltech.com/resources/intro-sc.html
- RS232 Tutorial:
  - http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html

- Peter Fleury Online:
  - Serial Port Interface for an AVR: http://homepage.hispeed.ch/peterfleury/avr-uart.html
  - UART Library: http://homepage.hispeed.ch/peterfleury/avr-software.html#libs

Terminale alternatywne dla programu HyperTerminal

- Br@y Terminal:
  - http://sites.google.com/site/braypp/terminal
- RealTerm:
  - http://realterm.sourceforge.net/
- TeraTerm:
  - http://www.ayera.com/teraterm/
- Minicom (systemy LINUX):
  - https://alioth.debian.org/projects/minicom/

Politechnika Łódzka

Instytut Elektroniki

Dziękuję za uwagę.