Systemy wbudowane

Wykład 6 - transmisje szeregowe: UART i pochodne

Przemek Błaśkiewicz

28 marca 2019

Komunikacja szeregowa

`http://websdr.org`

Rodzaje transmisji

simplex/sympleks

Komunikacja taka, że Rx i Tx są niewymienne: jednokierunkowa.

Rodzaje transmisji

simplex/sympleks

Komunikacja taka, że Rx i Tx są niewymienne: jednokierunkowa.

duplex/dupleks

Komunikacja, gdzie Rx i Tx zamieniają się rolami: dwukierunkowa.

Rodzaje transmisji

simplex/sympleks

Komunikacja taka, że Rx i Tx są niewymienne: jednokierunkowa.

duplex/dupleks

Komunikacja, gdzie Rx i Tx zamieniają się rolami: dwukierunkowa.
Półdupleks (half duplex): komunikacja jest naprzemienna.

Rodzaje transmisji

simplex/sympleks

Komunikacja taka, że Rx i Tx są niewymienne: jednokierunkowa.

duplex/dupleks

Komunikacja, gdzie Rx i Tx zamieniają się rolami: dwukierunkowa.
Półdupleks (half duplex): komunikacja jest naprzemienna.
Pełny dupleks (full duplex): komunikacja w obie strony jest niezależna

UART

UART

- Universal Asynchronous Receiver and Transmitter

UART

- Universal Asynchronous Receiver and Transmitter
- służy do przesyłania informacji bit po bicie (na jednej linii)

UART

- Universal Asynchronous Receiver and Transmitter
- służy do przesyłania informacji bit po bicie (na jednej linii)
- posiada bufory we/wy dla wygody szybszych transmisji

UART

- Universal Asynchronous Receiver and Transmitter
- służy do przesyłania informacji bit po bicie (na jednej linii)
- posiada bufory we/wy dla wygody szybszych transmisji
- odciąża procesor w zadaniu komunikacji

UART

- Universal Asynchronous Receiver and Transmitter
- służy do przesyłania informacji bit po bicie (na jednej linii)
- posiada bufory we/wy dla wygody szybszych transmisji
- odciąża procesor w zadaniu komunikacji
- *nie* zajmuje się poziomami logicznymi/napięciowymi na liniach

UART

- Universal Asynchronous Receiver and Transmitter
- służy do przesyłania informacji bit po bicie (na jednej linii)
- posiada bufory we/wy dla wygody szybszych transmisji
- odciąża procesor w zadaniu komunikacji
- *nie* zajmuje się poziomami logicznymi/napięciowymi na liniach
- RS-232, RS-485, IrDA, Bluetooth tryb SPP, modem

UART - wnętrzności

"Paczka danych":

UART - wnętrzności

"Paczka danych":

- bit START

UART - wnętrzności

"Paczka danych":

- bit START
- 5,6,7 lub 8 bitów DANYCH

UART - wnętrzności

"Paczka danych":

- bit START
- 5,6,7 lub 8 bitów DANYCH
- 1 bit PARZYSTOŚCI (opcjonalnie)

UART - wnętrzności

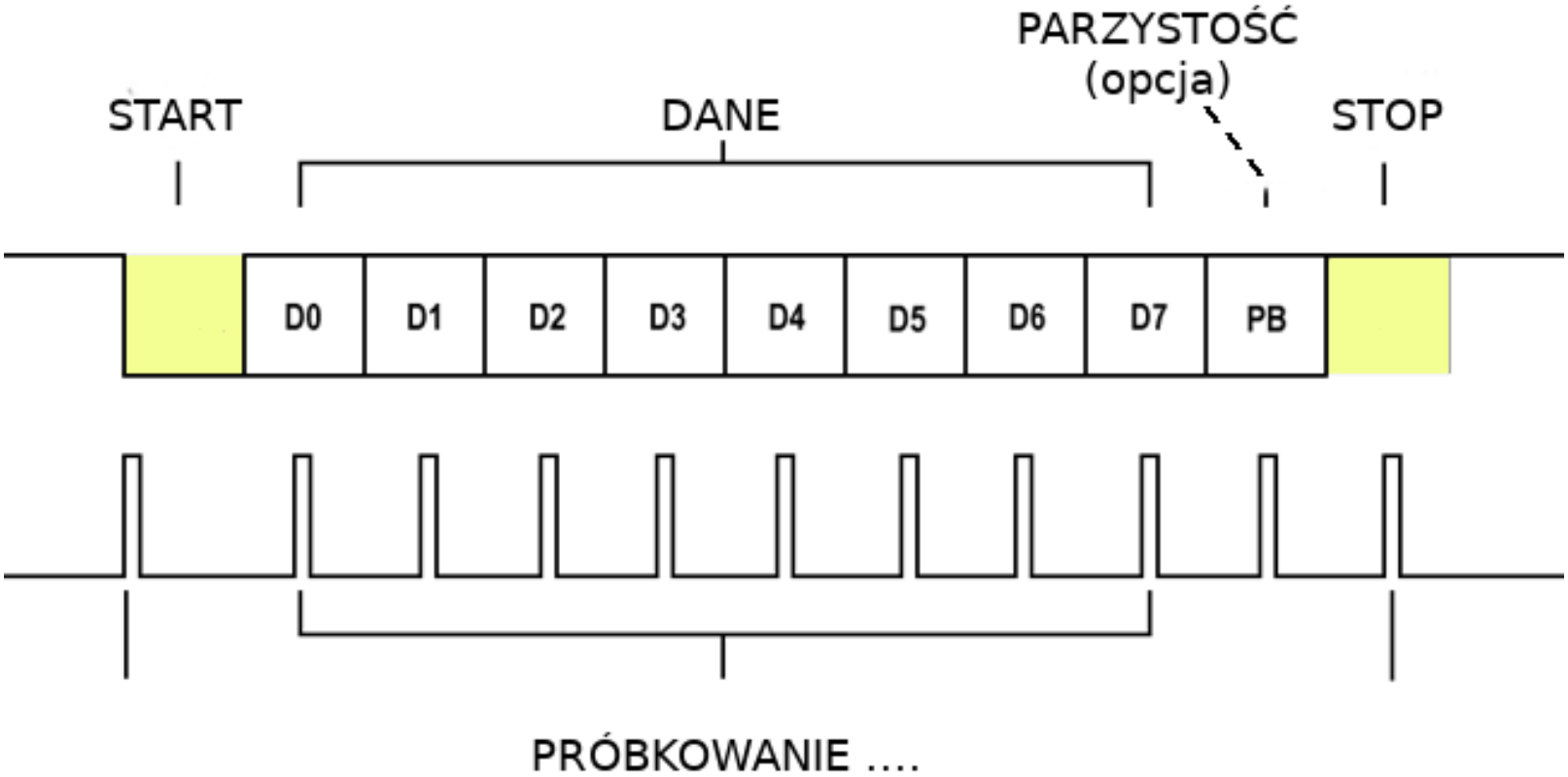
"Paczka danych":

- bit START
- 5,6,7 lub 8 bitów DANYCH
- 1 bit PARZYSTOŚCI (opcjonalnie)
- 1, 1.5 lub 2 bity STOP

UART - wnętrzności

"Paczka danych":

- bit START
- 5,6,7 lub 8 bitów DANYCH
- 1 bit PARZYSTOŚCI (opcjonalnie)
- 1, 1.5 lub 2 bity STOP



Bit parzystości parzystej i nieparzystej

Bit parzystości

Przyjmuje 0 dla **parzystej** liczby jedynek w słowie, 1 w p.p.

Bit parzystości parzystej i nieparzystej

Bit parzystości

Przyjmuje 0 dla **parzystej** liczby jedynek w słowie, 1 w p.p.

Bit nieparzystości

Przyjmuje 0 dla **nieparzystej** liczby jedynek w słowie, 1 w p.p.

Bit parzystości parzystej i nieparzystej

Bit parzystości

Przyjmuje 0 dla **parzystej** liczby jedynek w słowie, 1 w p.p.

Bit nieparzystości

Przyjmuje 0 dla **nieparzystej** liczby jedynek w słowie, 1 w p.p.

Bit parzystości parzystej i nieparzystej

Bit parzystości

Przyjmuje 0 dla **parzystej** liczby jedynek w słowie, 1 w p.p.

Bit nieparzystości

Przyjmuje 0 dla **nieparzystej** liczby jedynek w słowie, 1 w p.p.

```
function mxor(signal A : std_logic_vector)
return std_logic
is
  variable tmp : std_logic;
begin
  tmp := '0';
  for i in A'range loop
     tmp := tmp xor A(i);
  end loop;
  tmp := tmp xor '0';
  return tmp;
end mxor;
```

Transmisja UART

baud (Bd)

wym. bod – ilość *zmian* stanu medium na sekundę, związana z wysyłaniem symbolu kodowego (nie bitu informacji!).

Transmisja UART

baud (Bd)

wym. bod – ilość *zmian* stanu medium na sekundę, związana z wysyłaniem symbolu kodowego (nie bitu informacji!).
W przypadku gdy symbol kodowy jest równoważny bitowi baud = bitrate.

- bezczynna linia jest w stanie “Hi”

Transmisja UART

baud (Bd)

wym. bod – ilość *zmian* stanu medium na sekundę, związana z wysyłaniem symbolu kodowego (nie bitu informacji!).
W przypadku gdy symbol kodowy jest równoważny bitowi baud = bitrate.

- bezczynna linia jest w stanie "Hi"
- dane do wysłania wchodzą równolegle do UART $\rightarrow$ rejestr przesuwny

Transmisja UART

baud (Bd)

wym. bod – ilość *zmian* stanu medium na sekundę, związana z wysyłaniem symbolu kodowego (nie bitu informacji!).
W przypadku gdy symbol kodowy jest równoważny bitowi baud = bitrate.

- bezczynna linia jest w stanie "Hi"
- dane do wysłania wchodzą równolegle do UART $\rightarrow$ rejestr przesuwny
- wysyłany jest bit START (Lo)

Transmisja UART

baud (Bd)

wym. bod – ilość *zmian* stanu medium na sekundę, związana z wysyłaniem symbolu kodowego (nie bitu informacji!).
W przypadku gdy symbol kodowy jest równoważny bitowi baud = bitrate.

- bezczynna linia jest w stanie "Hi"
- dane do wysłania wchodzą równolegle do UART $\rightarrow$ rejestr przesuwny
- wysyłany jest bit START (Lo)
- odbiornik wykrywa stan Lo jeśli linia jest przez min 1/2 czasu trwania bitu w stanie Lo

Transmisja UART

baud (Bd)

wym. bod – ilość *zmian* stanu medium na sekundę, związana z wysyłaniem symbolu kodowego (nie bitu informacji!).
W przypadku gdy symbol kodowy jest równoważny bitowi baud = bitrate.

- bezczynna linia jest w stanie "Hi"
- dane do wysłania wchodzą równolegle do UART $\rightarrow$ rejestr przesuwny
- wysyłany jest bit START (Lo)
- odbiornik wykrywa stan Lo jeśli linia jest przez min 1/2 czasu trwania bitu w stanie Lo
- przez ustaloną ilość bitów do odbioru medium jest samplowane mniej więcej w środku przedziału

Transmisja UART

baud (Bd)

wym. bod – ilość *zmian* stanu medium na sekundę, związana z wysyłaniem symbolu kodowego (nie bitu informacji!).
W przypadku gdy symbol kodowy jest równoważny bitowi baud = bitrate.

- bezczynna linia jest w stanie "Hi"
- dane do wysłania wchodzą równolegle do UART $\rightarrow$ rejestr przesuwny
- wysyłany jest bit START (Lo)
- odbiornik wykrywa stan Lo jeśli linia jest przez min 1/2 czasu trwania bitu w stanie Lo
- przez ustaloną ilość bitów do odbioru medium jest samplowane mniej więcej w środku przedziału
- dane odebrane ładowane są do rejestru przesuwnego

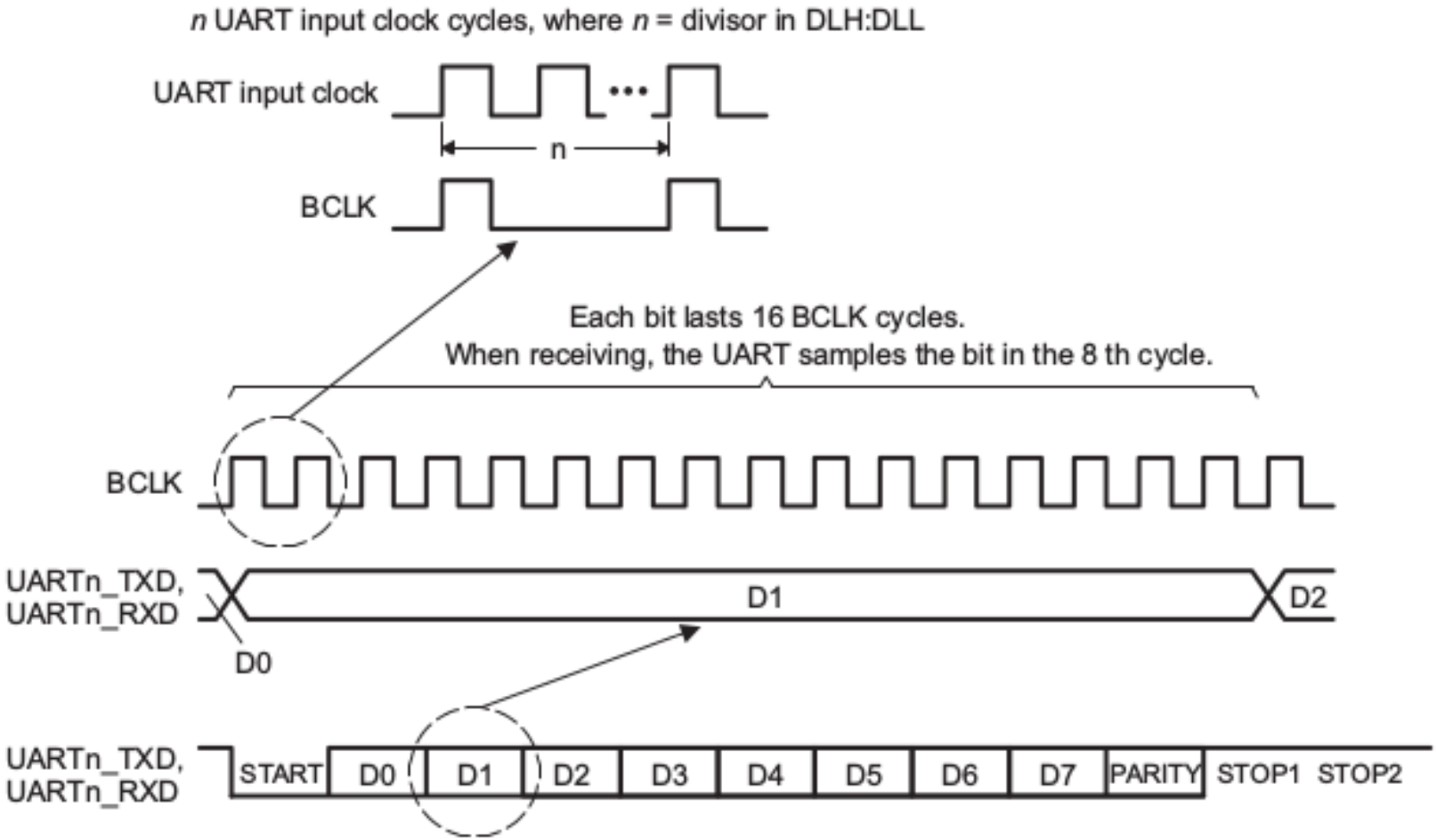
Transmisja UART

baud (Bd)

wym. bod – ilość *zmian* stanu medium na sekundę, związana z wysyłaniem symbolu kodowego (nie bitu informacji!).
W przypadku gdy symbol kodowy jest równoważny bitowi baud = bitrate.

- bezczynna linia jest w stanie "Hi"
- dane do wysłania wchodzą równolegle do UART $\rightarrow$ rejestr przesuwny
- wysyłany jest bit START (Lo)
- odbiornik wykrywa stan Lo jeśli linia jest przez min 1/2 czasu trwania bitu w stanie Lo
- przez ustaloną ilość bitów do odbioru medium jest samplowane mniej więcej w środku przedziału
- dane odebrane ładowane są do rejestru przesuwnego
- po odebraniu bitu STOP dane przesyłane są do rejestru odbiorczego

Transmisja UART



Transmisja UART

Table 2-1 Baud Rate Examples for 150-MHz UART Input Clock and 16× Oversampling Mode

Baud Rate	Divisor Value	Actual Baud Rate	Error (%)
2400	3906	2400.154	0.01
4800	1953	4800.372	0.01
9600	977	9595.701	-0.04
19200	488	19211.066	0.06
38400	244	38422.131	0.06
56000	167	56137.725	0.25
128000	73	129807.7	0.33
3000000	3	3125000	4.00

$\rightarrow$ 150MHz, div=3906, oversampl=16, 7 bitów danych, 1 parzystości

Transmisja UART

Table 2-1 Baud Rate Examples for 150-MHz UART Input Clock and 16× Oversampling Mode

Baud Rate	Divisor Value	Actual Baud Rate	Error (%)
2400	3906	2400.154	0.01
4800	1953	4800.372	0.01
9600	977	9595.701	-0.04
19200	488	19211.066	0.06
38400	244	38422.131	0.06
56000	167	56137.725	0.25
128000	73	129807.7	0.33
3000000	3	3125000	4.00

$\rightarrow$ 150MHz, div=3906, oversampl=16, 7 bitów danych, 1 parzystości

- $(150000000/3906)/16 = 2400.154)$

Transmisja UART

Table 2-1 Baud Rate Examples for 150-MHz UART Input Clock and 16× Oversampling Mode

Baud Rate	Divisor Value	Actual Baud Rate	Error (%)
2400	3906	2400.154	0.01
4800	1953	4800.372	0.01
9600	977	9595.701	-0.04
19200	488	19211.066	0.06
38400	244	38422.131	0.06
56000	167	56137.725	0.25
128000	73	129807.7	0.33
3000000	3	3125000	4.00

$\rightarrow$ 150MHz, div=3906, oversampl=16, 7 bitów danych, 1 parzystości

- $(150000000/3906)/16 = 2400.154)$
- START+7*DATA+PARITY+STOP $\rightarrow$ 10 bitów

Transmisja UART

Table 2-1 Baud Rate Examples for 150-MHz UART Input Clock and 16× Oversampling Mode

Baud Rate	Divisor Value	Actual Baud Rate	Error (%)
2400	3906	2400.154	0.01
4800	1953	4800.372	0.01
9600	977	9595.701	-0.04
19200	488	19211.066	0.06
38400	244	38422.131	0.06
56000	167	56137.725	0.25
128000	73	129807.7	0.33
3000000	3	3125000	4.00

$\rightarrow$ 150MHz, div=3906, oversampl=16, 7 bitów danych, 1 parzystości

- (150000000/3906)/16 = 2400.154)
- START+7*DATA+PARITY+STOP $\rightarrow$ 10 bitów
- 2400Bd $\rightarrow$ 2400 zmian (tu: bitów) na sekundę

Transmisja UART

Table 2-1 Baud Rate Examples for 150-MHz UART Input Clock and 16× Oversampling Mode

Baud Rate	Divisor Value	Actual Baud Rate	Error (%)
2400	3906	2400.154	0.01
4800	1953	4800.372	0.01
9600	977	9595.701	-0.04
19200	488	19211.066	0.06
38400	244	38422.131	0.06
56000	167	56137.725	0.25
128000	73	129807.7	0.33
3000000	3	3125000	4.00

$\rightarrow$ 150MHz, div=3906, oversampl=16, 7 bitów danych, 1 parzystości

- $(150000000/3906)/16 = 2400.154)$
- START+7*DATA+PARITY+STOP $\rightarrow$ 10 bitów
- 2400Bd $\rightarrow$ 2400 zmian (tu: bitów) na sekundę
- $2400 \frac{bit}{sek} / 10 \frac{bit}{paczka} = 240 \frac{paczka}{sek}$

UART i inne

- sygnały komunikacyjne: RTS, CTS (handshake)

UART i inne

- sygnały komunikacyjne: RTS, CTS (handshake)
- USART

UART i inne

- sygnały komunikacyjne: RTS, CTS (handshake)
- USART
  - synchronizuje Rx/Tx na podstawie ciągu danych

UART i inne

- sygnały komunikacyjne: RTS, CTS (handshake)
- USART
  - synchronizuje Rx/Tx na podstawie ciągu danych
  - w trakcie braku transmisji potrzebne są “pingi” (ASCII SYN 0x16)

UART i inne

- sygnały komunikacyjne: RTS, CTS (handshake)
- USART
  - synchronizuje Rx/Tx na podstawie ciągu danych
  - w trakcie braku transmisji potrzebne są “pingi” (ASCII SYN 0x16)
  - zwiększona przepustowość: brak START/STOP

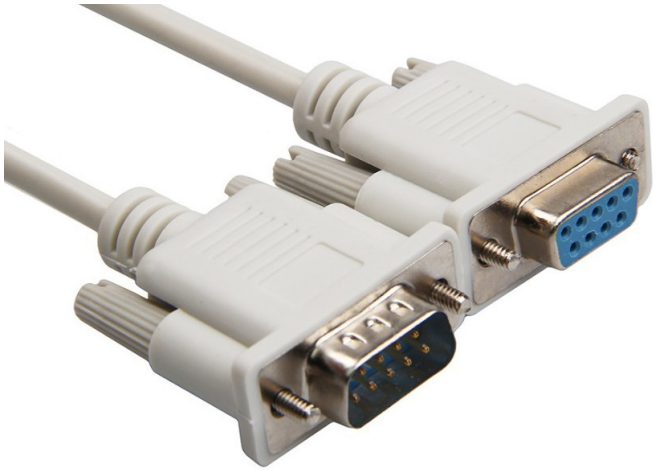
UART i inne

- sygnały komunikacyjne: RTS, CTS (handshake)
- USART
  - synchronizuje Rx/Tx na podstawie ciągu danych
  - w trakcie braku transmisji potrzebne są “pingi” (ASCII SYN 0x16)
  - zwiększona przepustowość: brak START/STOP
- DUART, OCTART

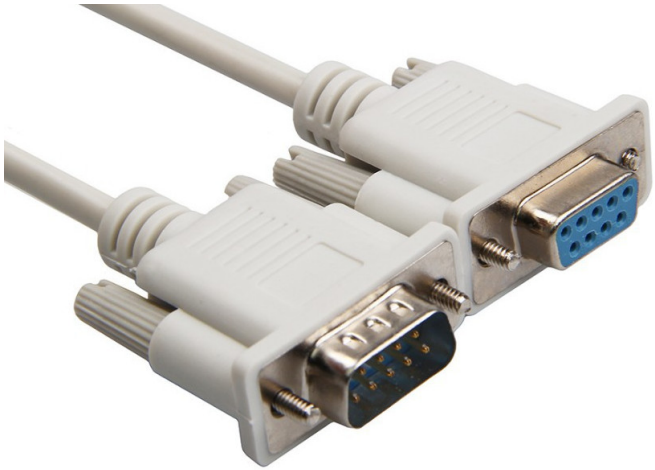
UART i inne

- sygnały komunikacyjne: RTS, CTS (handshake)
- USART
  - synchronizuje Rx/Tx na podstawie ciągu danych
  - w trakcie braku transmisji potrzebne są “pingi” (ASCII SYN 0x16)
  - zwiększona przepustowość: brak START/STOP
- DUART, OCTART
- bit-banging - software serial

RS-232

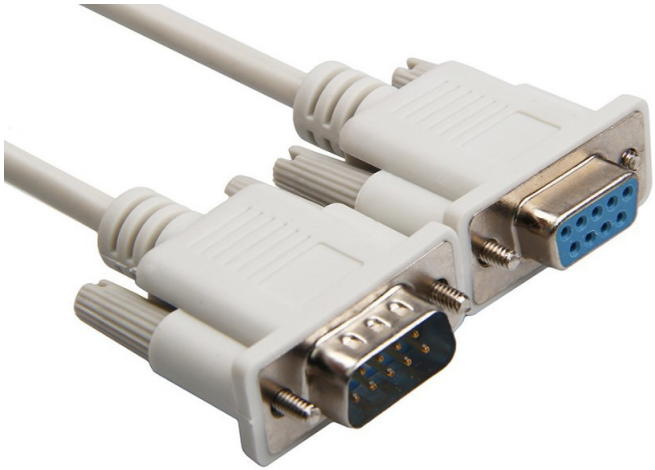


RS-232



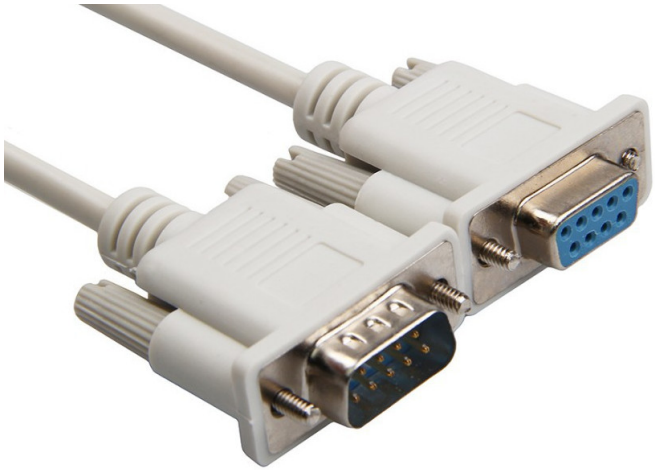
- Jest to *standard połączenia* urządzeń (nazwy styków, poziom sygnałów)

RS-232



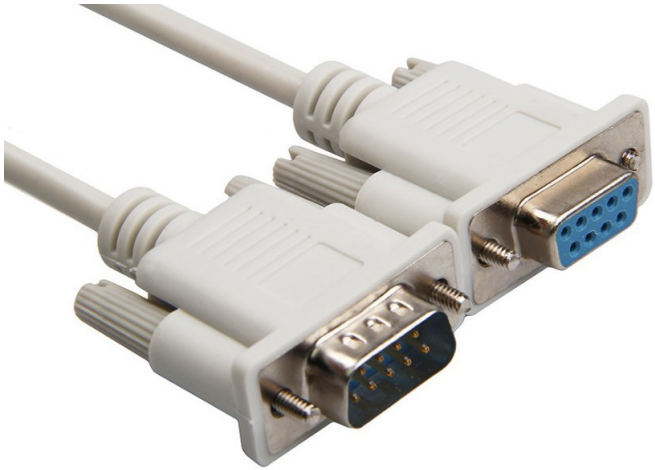
- Jest to *standard połączenia* urządzeń (nazwy styków, poziom sygnałów)
- Logiczne **1** jako napięcia od -3 do -15V, logiczne **0** jako 3 - 15V

RS-232



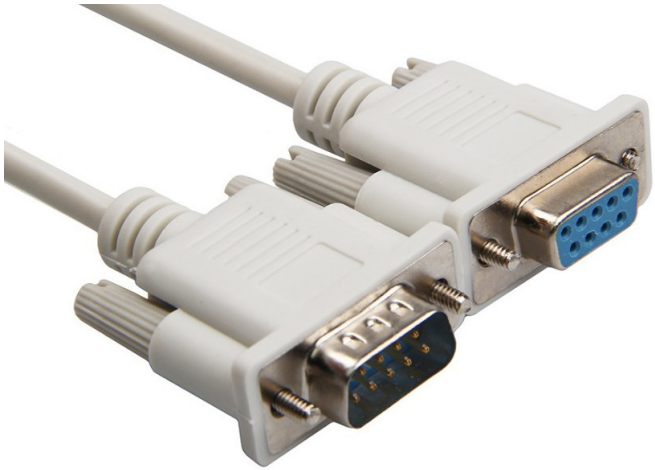
- Jest to *standard połączenia* urządzeń (nazwy styków, poziom sygnałów)
- Logiczne **1** jako napięcia od -3 do -15V, logiczne **0** jako 3 - 15V
- Typowo $\pm$ 5 V, $\pm$ 10 V, $\pm$ 12 V, $\pm$ 15 V

RS-232



- Jest to *standard połączenia* urządzeń (nazwy styków, poziom sygnałów)
- Logiczne **1** jako napięcia od -3 do -15V, logiczne **0** jako 3 - 15V
- Typowo $\pm$ 5 V, $\pm$ 10 V, $\pm$ 12 V, $\pm$ 15 V
- Potrzebne konwertery napięć!

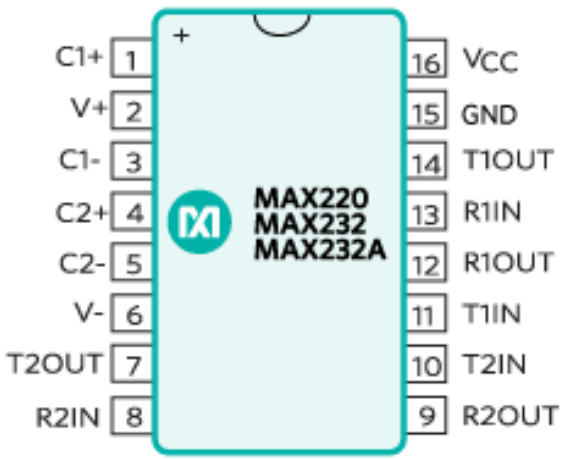
RS-232



- Jest to *standard połączenia* urządzeń (nazwy styków, poziom sygnałów)
- Logiczne **1** jako napięcia od -3 do -15V, logiczne **0** jako 3 - 15V
- Typowo $\pm$ 5 V, $\pm$ 10 V, $\pm$ 12 V, $\pm$ 15 V
- Potrzebne konwertery napięć!
- Wykorzystywane konwertery RS-232 $\leftrightarrow$ USB

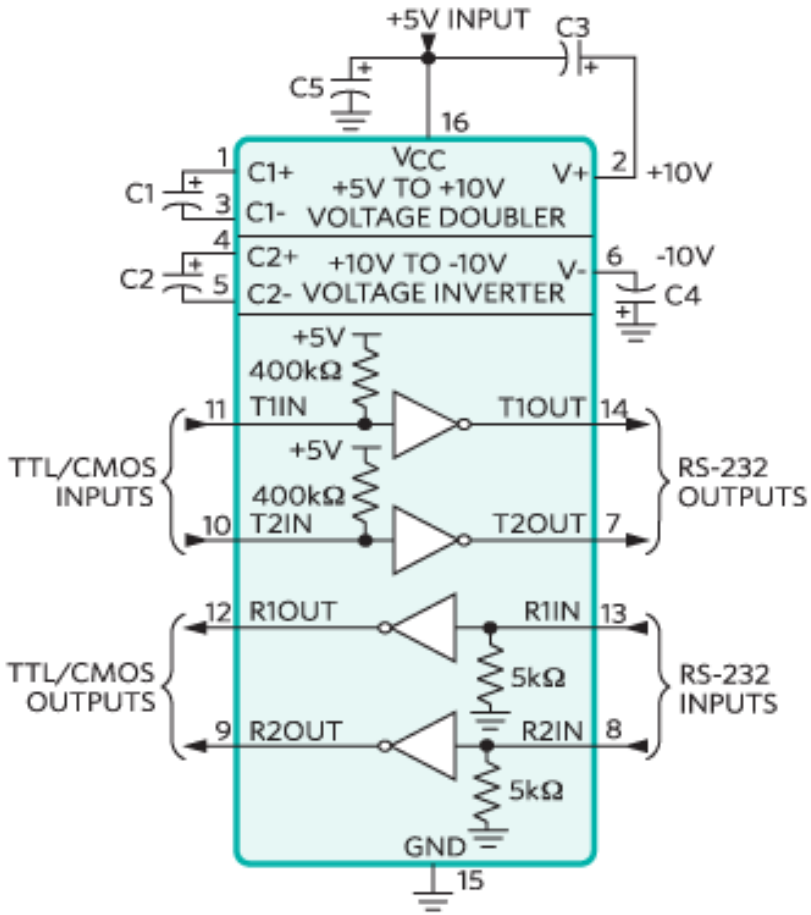
Rodzina MAXów

TOP VIEW

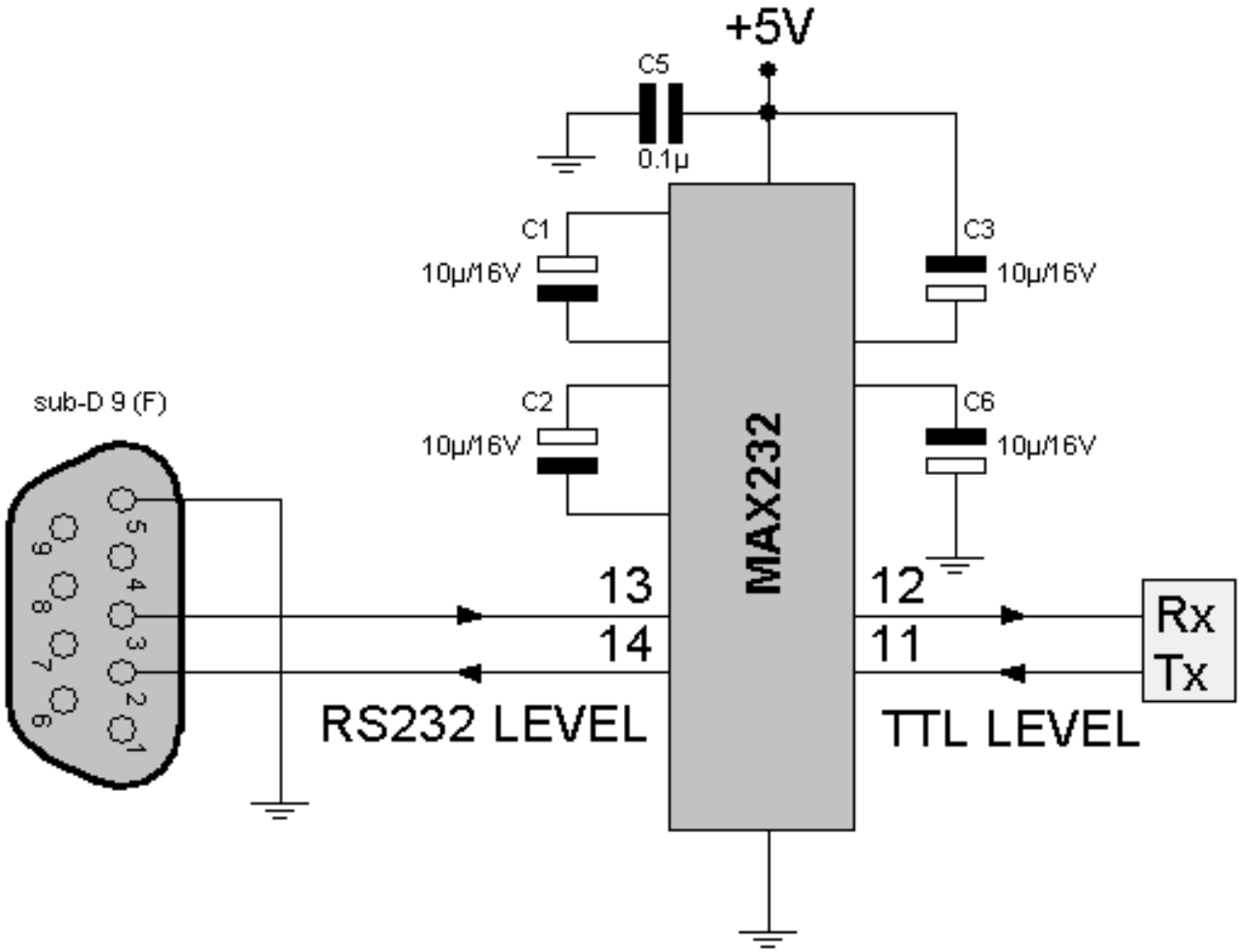


DIP/SO

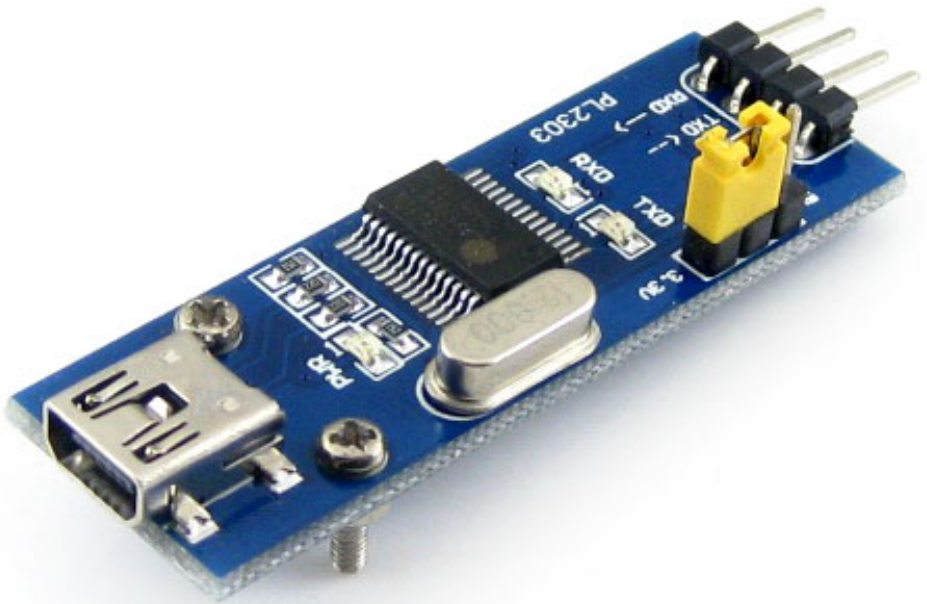
CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	0.047	0.33	0.33	0.33	0.33
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1



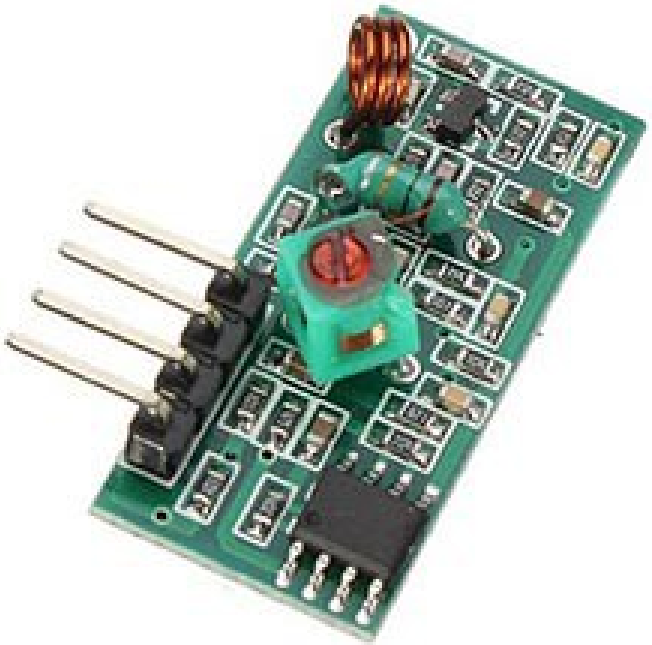
Rodzina MAXów



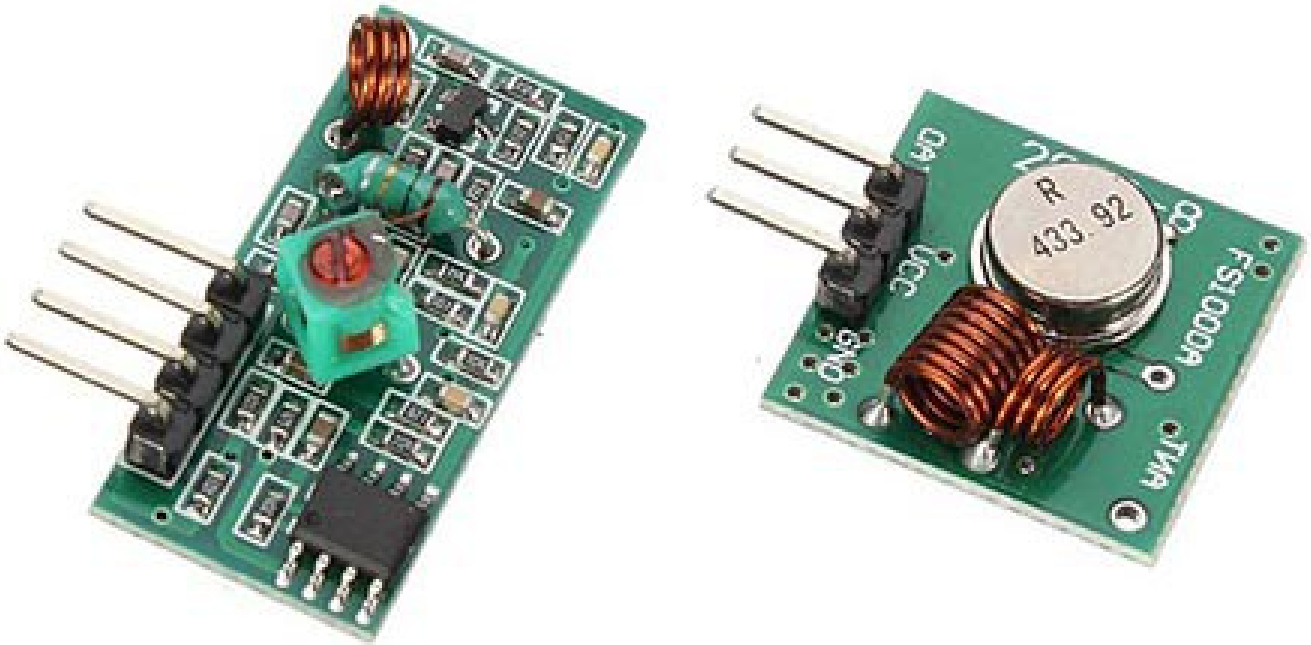
PL2302



Inne protokoły... SPI, I^2C,one-wire, oraz kontrolery w następnych odcinkach




R
433.92
GND
VCC
DATA
ANT



- Arduino + RF-433 + biblioteka RadioHead
- SDR: RTL8232U + gqrx (ew. GnuRadio)
- analizator stanów logicznych Saleae

Rzeczy do zapamiętania

- rodzaje transmisji, Bd,
- UART: tryby, działanie, ramka danych
- “radzenie sobie” z RS-232