Systemy wbudowane

Wykład do samodzielnego opracowania - transmisje szeregowe: UART i pochodne

Przemek Błaśkiewicz

25 marca 2020

Komunikacja szeregowa

Informacja przesyłana jest pojedynczym kanałem znak po znaku *ale* znak informacji przesyłany może być za pomocą wielu zdarzeń (np. impulsów).

Wszyscy znamy alfabet Morse'a, można go też posłuchać:
np. `http://websdr.ewi.utwente.nl:8901` okolice 3.510 MHz
Więcej słuchania na `http://websdr.org`

Rodzaje transmisji

simplex/sympleks

Komunikacja taka, że Rx (odbiór) i Tx (nadawanie) 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 przebiega niezależnie.

♠ jakie urządzenia korzystają z tych rodzajów transmisji?

UART

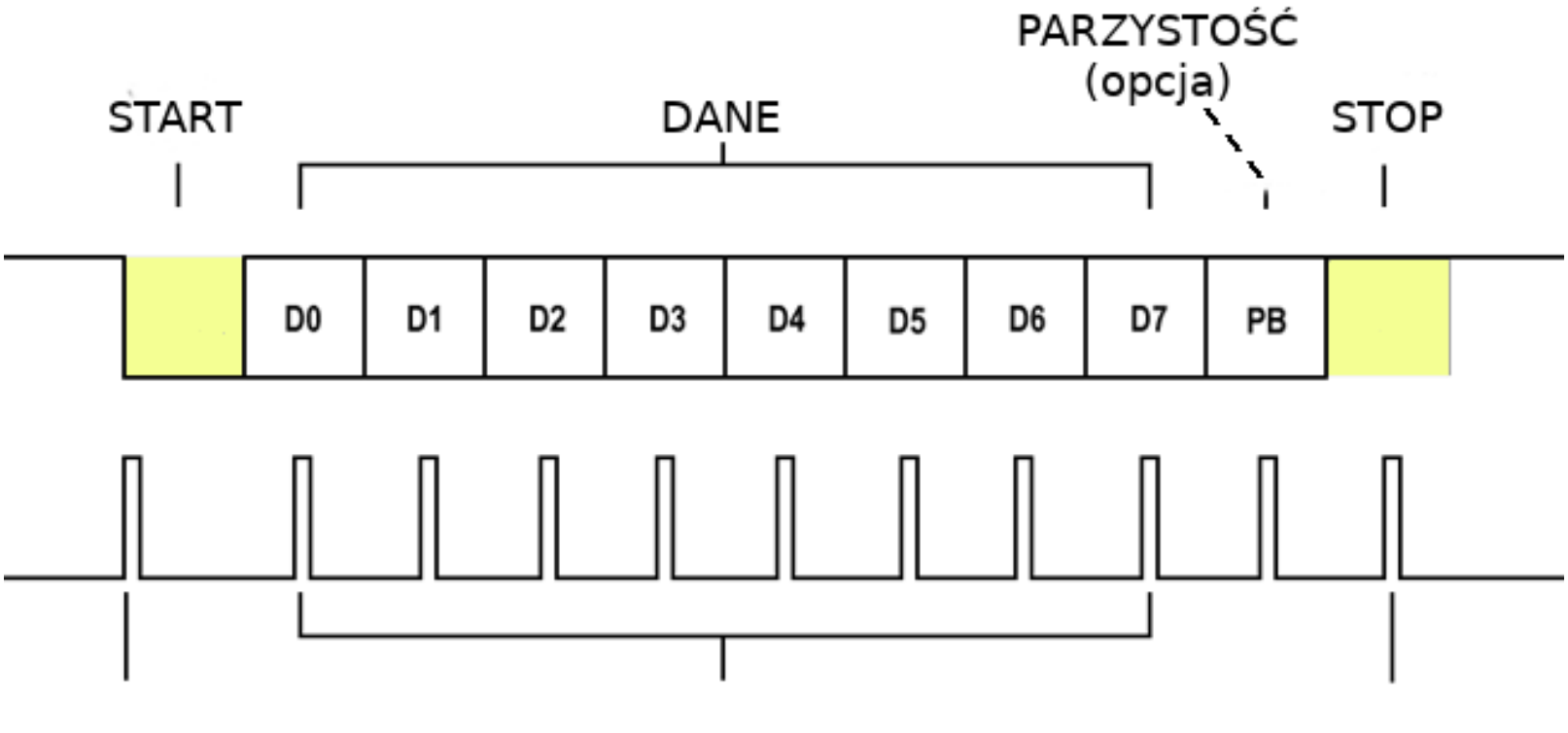
- Universal Asynchronous Receiver and Transmitter
- Służy do przesyłania informacji bit po bicie (na jednej linii)
- Zakłada wykorzystanie buforów we/wy dla wygody szybszych transmisji i odciążżenia procesora:
  - Procesor ładuje do bufora i daje komendę “ślij”.
  - Odebrane bity trafiają do rejestru, po skompletowaniu procesor dostaje informację (przerwanie?) “gotowe”.
- *nie* zajmuje się poziomami logicznymi/napięciowymi na liniach
- Przykłady urządzeń: RS-232, RS-485, IrDA, Bluetooth tryb SPP, modemy

UART - wnętrzności

Przesyłane komunikaty dzielone są na "paczki danych"(znaki), a te opakowywane są w strukturę:

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

  ↪ 1.5 bitu w znaczeniu *czas trwania 1 i pół bitu*.



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.

♠ Jaką parzystość liczy ten kod VHDL?

```
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;
```

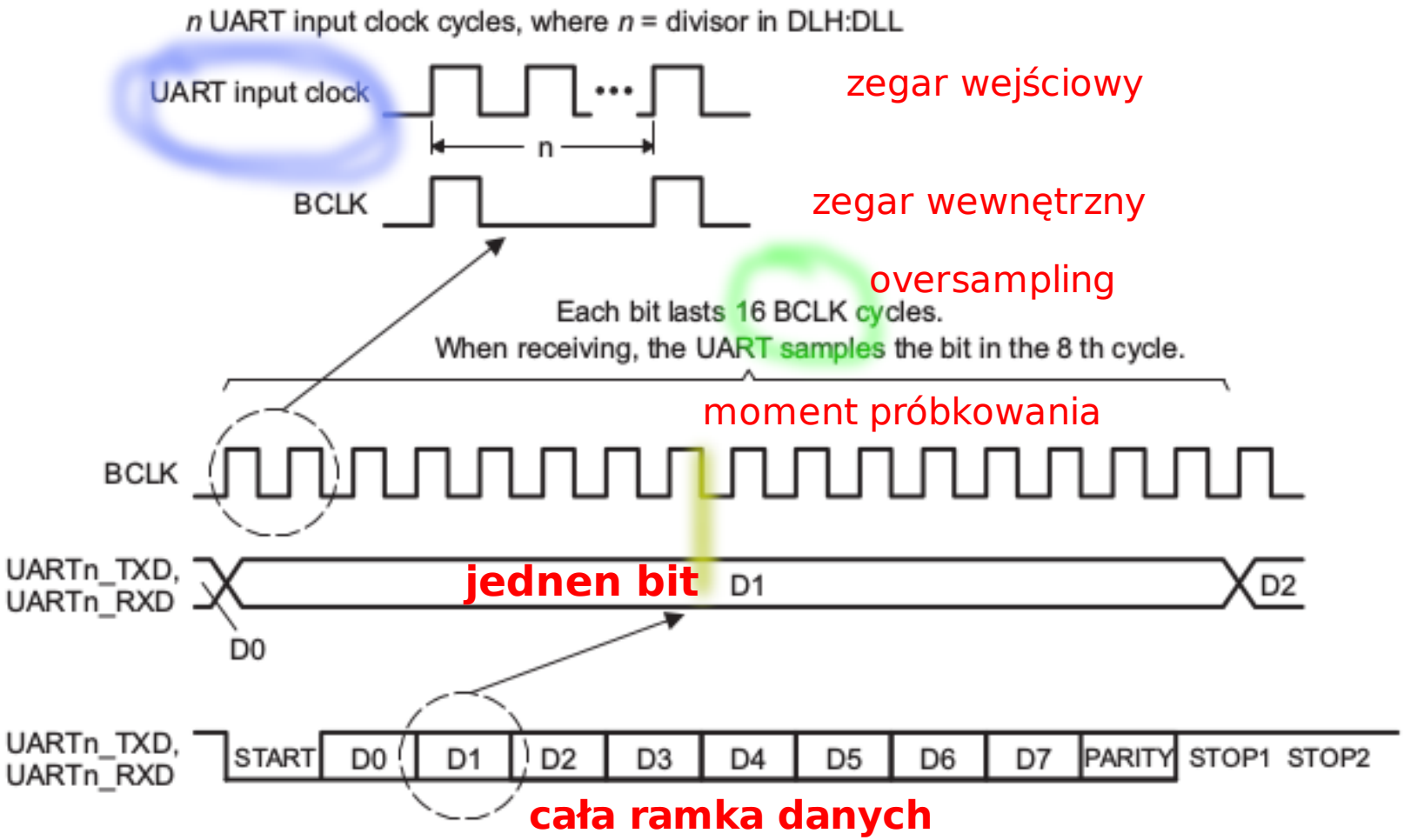
Transmisja UART

baud (Bd)

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

- bezczynna linia jest w stanie "High"
- dane do wysłania wchodzą (np. równolegle) do Tx rejestru przesuwnego UART
- wysyłany jest bit START (Lo)
- odbiornik wykrywa stan Lo jeśli linia jest przez min 1/2 czasu trwania bitu w stanie Lo
- przez ustaloną liczbę bitów do odbioru medium jest samplowane *mniej więcej* w środku przedziału
  $\hookrightarrow$ ta dowolność pozwala na lekkie niedopasowanie zegarów odbiornika i nadajnika
- dane odebrane ładowane są do rejestru przesuwnego
- po odebraniu bitu STOP dane przesyłane są do rejestru Rx i z

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

♠ **Przykład:** zegar wejściowy: 150MHz, oversampling=16, 7 bitów danych, 1 bit parzystości, n=3906. *Ile takich 7-bitowych informacji można wysłać na sekundę?*

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

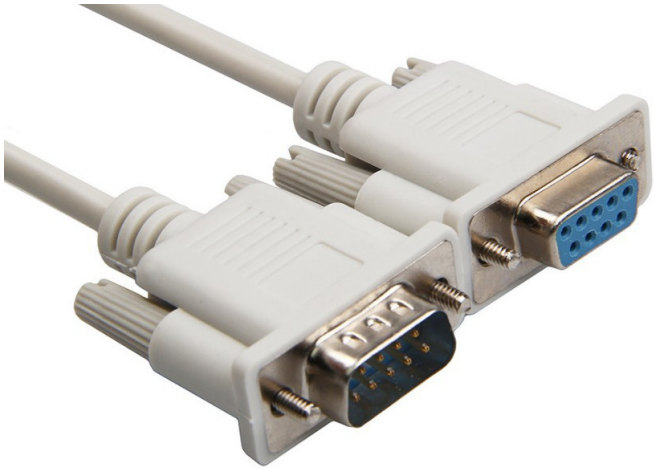
Transmisja UART

Możliwe prędkości dla UART są "dziwne", ale mają uzasadnienie. Dobrano je tak, żeby dla typowych prędkości taktowania można było zagwarantować możliwie duże zsynchronizowanie transmisji pomiędzy urządzeniami o różnych zegarach.
Dla powyższego przykładu, jeśli urządzenie odbiorcze taktowane jest zegarem 8 MHz, to przy tych samych ustawieniach oversamplingu musi być $n = 208\frac{1}{3} \rightarrow 208$, co daje baudrate 2403,8, czyli 0.39% odchyłki od 2400.154 bps. ♠
Zatem dwa skrajnie różne co do szybkości urządzenia są w stanie wymienić informacje bez ustalania wspólnego zegara – "rozjadą się" co najwyżej o mniej niż jedną setną trwania przesyłu jednego bitu.

UART i inne

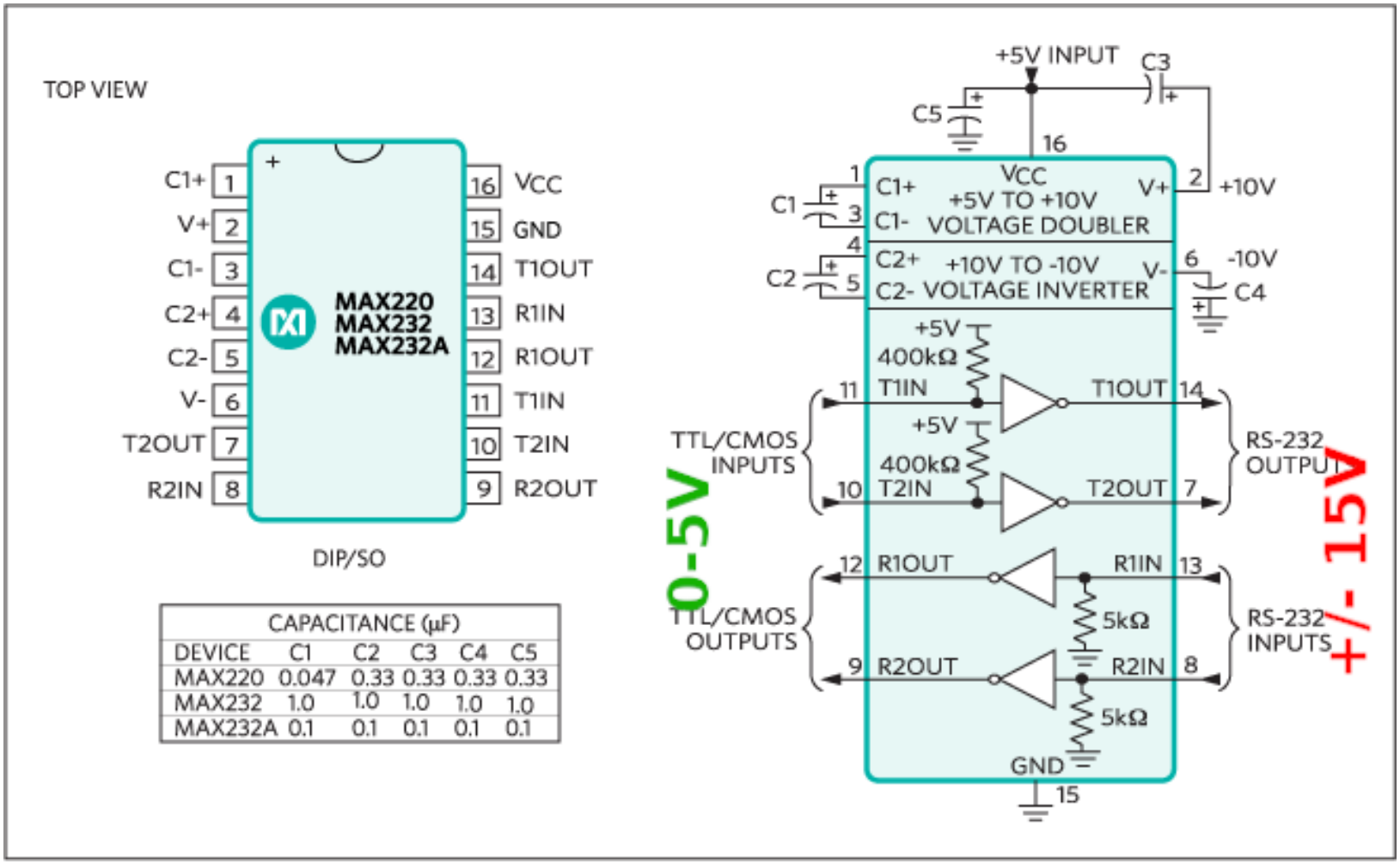
- sygnały komunikacyjne: RTS, CTS (handshake)
  - RequestToSend: linia ustawiana przez Tx w momencie gotowości do wysyłania danych;
  - ClearToSend: linia ustawiana w odpowiedzi przez Rx – gotowość do odbioru danych;
  - ♠ `https://en.wikipedia.org/wiki/RS-232#Data_and_control_signals`
- USART
  - synchronizuje Rx/Tx na podstawie ciągu danych
  - w trakcie braku transmisji potrzebne są "pingi" (ASCII SYN 0x16)
  - zwiększona przepustowość: brak START/STOP
- DUART, OCTART – zwiększenie liczby linii transmisyjnych i układów sterujących
- bit-banging - `SoftwareSerial` z Arduino
  ↪ zmiany stanu linii danych generowane przez sam procesor, a nie specjalny układ (zajmuje procesor, często zabiera też licznik/timer do podzielenia częstotliwości tak, by zapewnić odpowiedni baudrate).

RS-232



- Jest to *standard połączenia* urządzeń (nazwy styków, poziom i wartościowanie sygnałów)
- Logiczne **1** to napięcia między -3 a -15V, logiczne **0** między 3 a 15V
  $\hookrightarrow$ typowo $\pm$ 5 V, $\pm$ 10 V, $\pm$ 12 V, $\pm$ 15 V
- Ponieważ procesory/mikrokontrolery działają zazwyczaj z napięciami 0-5V (TTL) lub 0-3.3V (LVTTL), potrzebne są konwertery napięć!
- A do podłączania układów UART np. do komputera wykorzystuje się konwertery RS-232 $\leftrightarrow$ USB

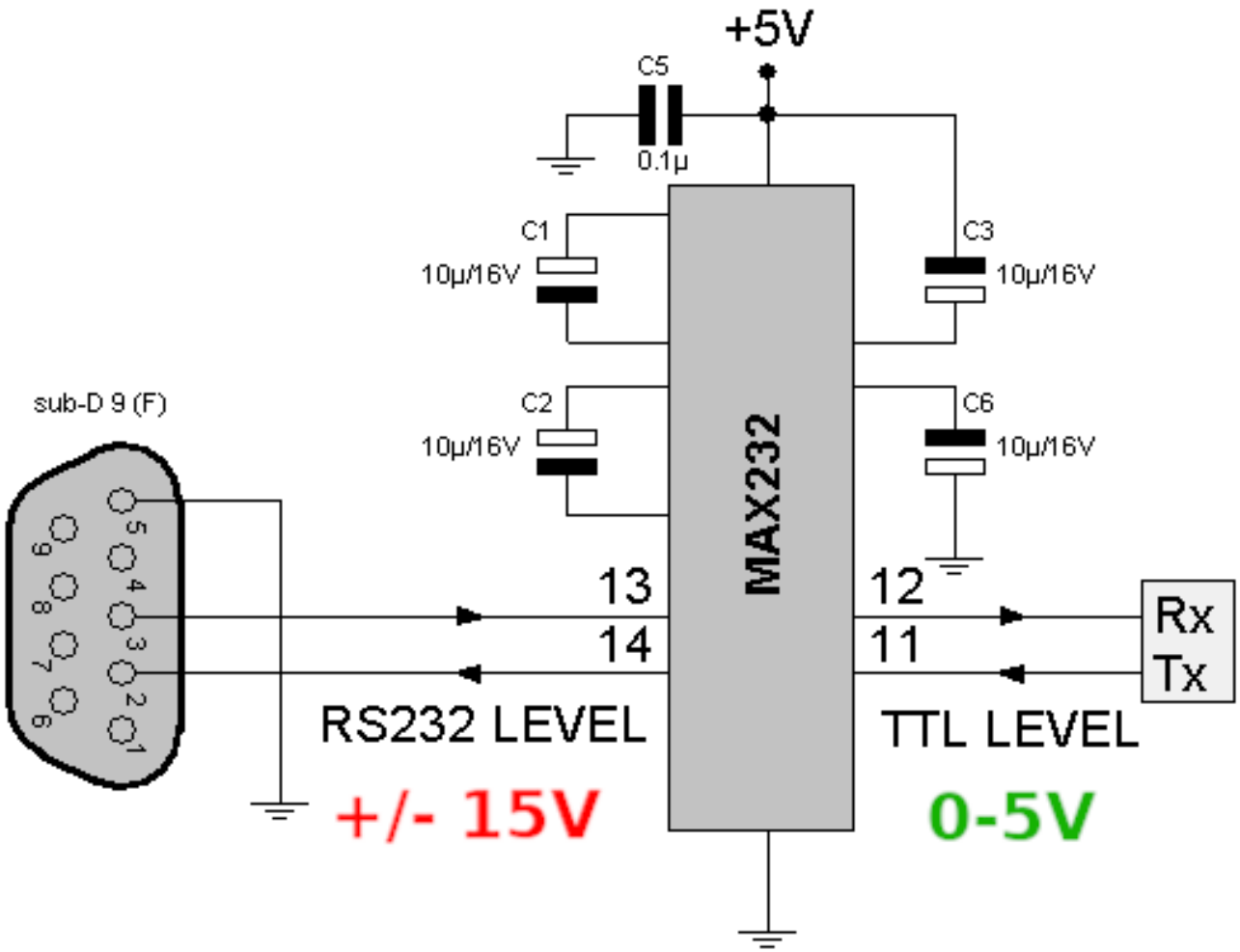
Rodzina MAXów



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

Z noty katalogowej

Rodzina MAXów



Podstawowy schemat połączeń: tylko linie TxD, RxD i GND

PL2303

Konwerter USB–RS-232 na PL2303. Umożliwia ustawienie poziomu logiki TTL, pojawia się w systemie jako “port COM” (ttyUSBx lub ttyACMx na linuksach).
Często wbudowany w różne urządzenia (np. ręczne GPSy), przeprogramowany tak, żeby przedstawiał się jako właśnie to urządzenie.
Są wersje obsługujące jednocześnie dwa urządzenia UART.

- Inne protokoły... SPI, I^2C,one-wire, oraz kontrolery w następnych odcinkach
- ♠ `https://www.youtube.com/watch?v=E9CfJWIe1_o` ↪ (5 minut - proszę obejrzeć!)
- ♠ `https://www.youtube.com/watch?v=avWdWivJKDw` ↪ (bardziej rozbudowane, ale polecam ten kanał!)