

Monitor portu RS-232

Zadanie: Należy zaprojektować moduł odbiornika danych w standardzie RS-232 dla następujących parametrów transmisji: 8 bitów danych, bez bitu parzystości, jeden bit stopu, szybkość transmisji 9600bps bez sprzętowej kontroli przepływu. Moduł należy połączyć z modułem display z poprzedniego zadania tak, aby odbiornik RS232 po otrzymaniu danych wyświetlał je na wyświetlaczu LED w postaci heksadecymalnej (2 cyfry hex). Do wyświetlania odebranego kodu wykorzystać dowolnie wybrane dwie kolejne cyfry wyświetlacza.

W użyciu jest prawa połowa (4 cyfry) wyświetlacza 7-segmentowego (lewa połowa może także coś wyświetlać – ignorujemy to).

Wejścia i wyjścia układu:

clk_i - zegar 100MHz,

rst_i - reset asynchroniczny (przycisk BTNR),

RXD_i - wejście danych RS232,

led7_an_o – wyjście sterujące anodami wyświetlaczy LED.

led7_seg_o – wyjście sterujące segmentami wyświetlaczy LED.

Należy wykonać symulację funkcjonalną oraz zweryfikować układ praktycznie poprzez zaprogramowanie płytki testowej.

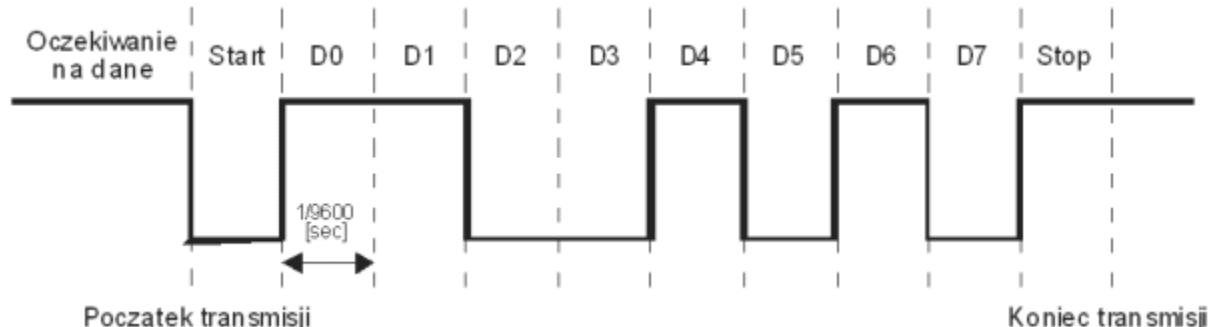
Aby wysyłać dane na płytke należy uruchomić program Putty, wybrać połaczenie „Serial”, wpisać prędkość (9600) oraz port: COMx (x to numer portu, należy go odczytać w programie „device manager” (program ten można łatwo znaleźć na zdalnym komputerze za pomocą klawisza szukania „lupa”) – pod pozycją „Ports (COM&LPT)” należy znaleźć urządzenie o nazwie „USB serial Port” – numer portu będzie podany obok). Należy jeszcze włączyć echo lokalne, aby znaki wysyłane do płytki były widoczne na ekranie – trzeba kliknąć w kategorię „Terminal” i zaznaczyć opcję „Force on” dla „Local echo”. Potem kliknąć w klawisz „Open” w celu otwarcia połączenia.

Naciśnięcie klawisza na klawiaturze powinno spowodować wyświetlenie jego kodu ASCII na wyświetlaczu LED.

Nadawanie i odbiór sygnałów w standardzie RS232

Nadawanie i odbiór sygnałów w standardzie RS232 odbywa się w sposób szeregowy, oddzielnie na dwóch liniach w kierunkach do i od urządzenia. W czasie braku transmisji sygnał na danej linii jest w stanie wysokim. Rozpoczęcie transmisji inicjowane jest przez tzw. bit startu będący "0" logicznym, który powinien trwać przez okres równy odwrotności prędkości transmisji, w naszym przypadku wynoszący 1/9600 [sekund]. Wszystkie następne informacje są przesyłane z identycznym okresem trwania. Następnie nadawane są szeregowo dane począwszy od najmniej znaczącego bitu aż do bitu najbardziej znaczącego (**D0-D7**). Później występuje bit parzystości, będący operacją logiczną **XOR** na danych **D0-D7**. Bit parzystości jest opcjonalny i w przypadku niniejszego zadania

nie występuje. Zakończenie transmisji sygnalizowane jest bitami stopu, w ilości zazwyczaj od 1 do 2. Przykład sygnału, zgodnego z wymaganiami zadania, przenoszącego kod **01010011** przedstawiony jest na poniższym rysunku.



Rys. 1 Przykładowa transmisja kodu 01010011 przez RS-232

Informacje dodatkowe o standardzie RS232:

<http://www.fizyka.umk.pl/~ptarg/labview/folie/RS232.pdf>
<http://pl.wikipedia.org/wiki/RS-232>

Minimalne wymagania dotyczące symulacji: wykonać reset na początku symulacji, podczas symulacji wysłać daną z przykładu (Rys. 1) trzy razy, za pierwszym razem z prędkością nominalną (9600 bps), za drugim z prędkością mniejszą o 4% od nominalnej, za trzecim z prędkością większą o 4% od nominalnej.

Uwaga! W projekcie musi być zawarty przerzutnik synchronizujący sygnał RXD_i z zegarem!

Fragment głównego pliku projektowego VHDL z deklaracją sygnałów:

```
entity top is
    Port ( clk_i : in STD_LOGIC;
            rst_i : in STD_LOGIC;
            RXD_i : in STD_LOGIC;
            led7_an_o : out STD_LOGIC_VECTOR (3 downto 0);
            led7_seg_o : out STD_LOGIC_VECTOR (7 downto 0));
end top;
```

Plik z ograniczeniami projektowymi dla płytka Nexys-A7 (układ FPGA xc7a100tcsg324-1): [isp4s.xdc](#)