Communication Methods in Digital Systems Laboratory

Serial Asynchronous Data Transmission

Introduction

This laboratory is intended to illustrate serial communication using different methods of data transmission starting from asynchronous using different concepts of synchronous transmission including Manchester encoding and bit stuffing.

The experiments will be started from asynchronous transmission. The problem requires both hardware implementation and respective software implementation to complete the task of communication between the host computer and the device.

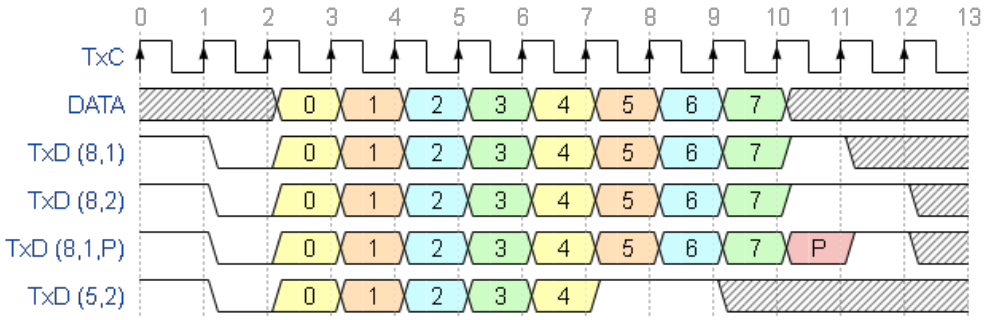

Fig. 1. Exemplary frames of different character length, stop field duration and optional data protection bit (parity)

The asynchronous serial data frame is shown in Fig. 1. The data frame begins with an obligatory start bit followed by the character whose length is in the range from 5 to 8 bits. Next, the character field is followed by the optional protection field. There is not specified method of calculation. Using a parity calculated as simply as just an XOR result of all bits of the character allows extending the Hamming code distance to 1. This enables the detection of single errors in the transmitted frame. Finally, the data frame is terminated and separated from the next one with stop bits. The duration of the stop space (gap) can be selected as 1, 1.5, or 2-bit signaling time.

The transmission of the data frame is a general serializing of the data including additional obligatory fields. The essential problem of each data transfer is the problem of correct data reception.

The essential problem of the correct reception is synchronization to the received data stream. The reception idea is shown in Fig. 2. The reception process is triggered by the falling edge of the RxD line. In the synchronous circuits, it is observed as a value difference between two consecutive samples. The receiver utilizes an oversampling

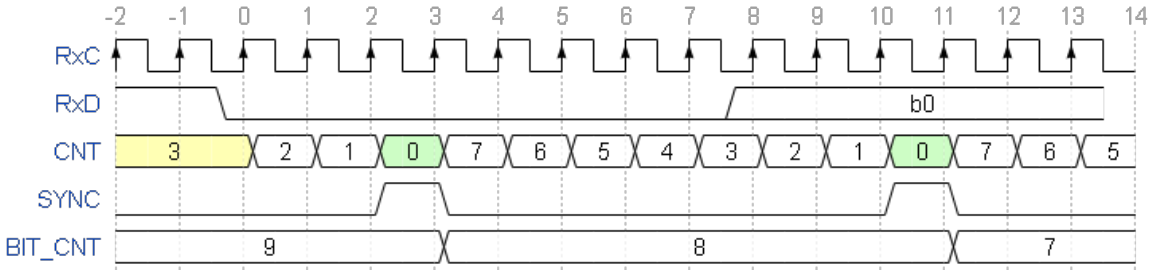

Fig. 2. Principle of the serial asynchronous data reception
with an 8-time oversampling

approach. The line is sampled typically with 16-time oversampling. The Fig. 2 shows the 8-time oversampling. In order to ensure the sampling point falls in the middle of the signalling period it is delayed by the counter. In the considered case it is a modulo 8 counter. The modulo number should be equal to the oversampling frequency. Initially, the counter is loaded with the value 3. The down-counting process starts when the beginning of the start bit is detected. In the middle of the bit, the SYNC signal is asserted for one clock period. As the SYNC is supposed to trigger synchronous operation the triggering point (a following clock period) is located in the middle of the signaling period. It should be noted, that the uncertainty of sampling is equal to one clock period. In the worst case, there is a margin of 3 clock periods to the boundary of bit signalling time.

The SYNC signal is used for sampling the TxD line value and updating the state of the bit counter (BIT_CNT). When the bit counter reaches the value of 0 the reception of the frame is complete. It should be noted that it allows checking for the STOP bit state which should be 1.

Asynchronous data transmission tasks

1. Design a synthesizable asynchronous receiver and transmitter for 8-bit characters and 1 stop-bit. Implement a transmission clock synthesizer based on direct digital synthesis (phase accumulator).
    a. Describe the transmitter and receiver and verify its properties in the simulation process. The receiver should notify about completing the reception and handle frame error (when 0 is received in place of STOP bit).
    b. Check using simulation the maximal frequency error for the designed receiver that allows for the correct reception of characters.
2. Implement the receiver and transmitter in the evaluation board. Ensure transmission of preprogrammed characters using switches. Display the

received character on the LEDs and the frame error flag. Implement separate clock generators for the receiver and transmitter. One of the generators should be programmed using slide switches. Link the TXD and RXD lines externally and check the limits of baud rate generation that allows for correct reception.

3. Implement a unit that is able to handle a simple protocol for controlling LEDs and push buttons state. Write a C program using the FTDI library that enables communication with the developed protocol. Use a signalling speed of 921600 bps in implementation. All commands are started by: colon. The summary is given below:
    a. Attention command ":!" should return the presence answer by unit ":$"
    b. The command ":?" should return the human-readable version string e.g. "IO by xx ver.1.00" The message should be stored in ROM, implemented using the case statement.
    c. The LED is controlled by the command ":L<n><s>" where n is the LED number while s is state of the LED 0 (off) and 1 (on).
    d. The pushbutton state must be reported immediately on change by sending the following frame ":P<n><s>" where: <n> is the button number and <s> is state 0 (released) or 1 (depressed).