ESIstream

The **E**fficient **S**erial **I**nterface

Implementation Note 02: Example of Architecture

Version 1.0

ESIstream is an Open high efficiency serial interface protocol based on 14b/16b encoding.

Its main benefits are low overhead and ease of hardware implementation.

TABLE OF CONTENT

FIGURE TABLE

EXAMPLE OF ARCHITECTURE

The ESIstream protocol is simple and efficient; it does not require a lot of logic resource to implement. This application note presents example of architecture that can be used to implement a transmitter and a receiver using the ESIstream protocol.

1.1 Transmitter (TX)

The transmitter is composed of 2 main parts, a logic based part that encodes the data and a hardware part that is the high speed transceiver.

The diagram below show a high level example of the composition of the transmitter:

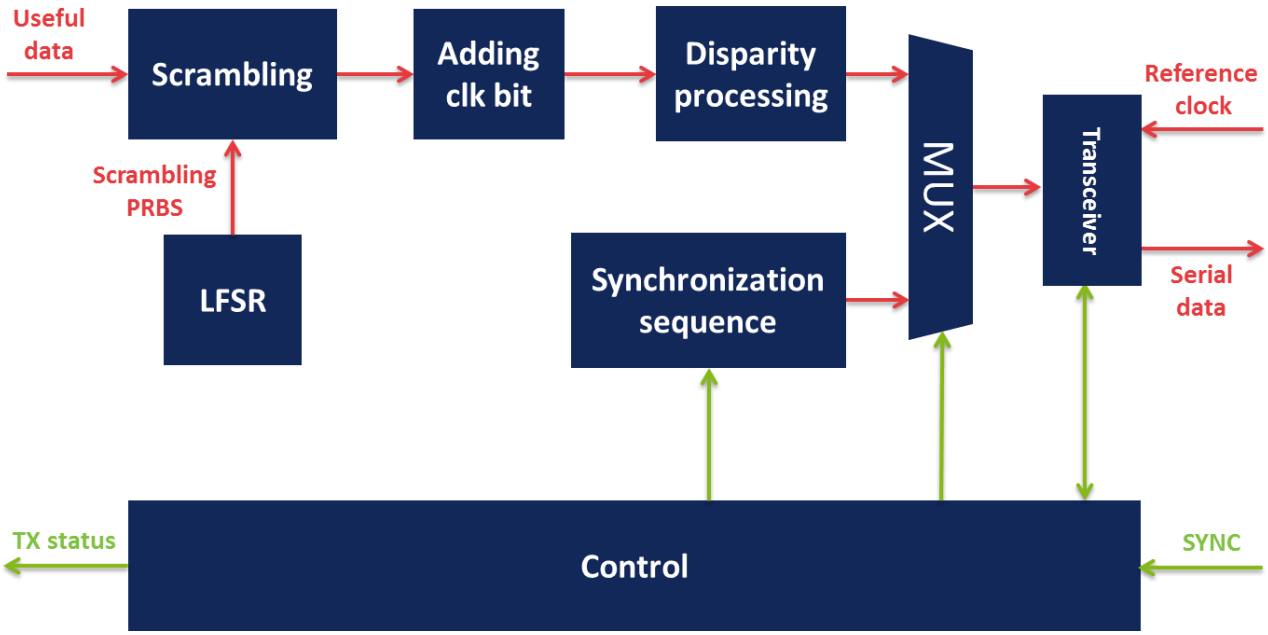

Figure 1: TX bloc diagram

The digital useful data are fed through the 1st stage of the transmitter: the scrambler. This unit scrambles the data with a pseudo-random binary sequence generated by an LFSR (Linear Feedback Shift Register). The second step consists of adding the overhead clock bit of the ESIstream protocol that toggles. Following this, the disparity is processed to ensure a DC balanced transmission.

This encoding is muxed with the synchronization sequence used when initializing the serial interface to synchronize the RX with the TX.

Whereas the serial interface is synchronizing or is in normal operating mode, the transceiver copes with the serialization of the data. It requires a slow reference clock which is used to generate the fast clock that clocks the high speed data.

The control part has multiple objectives. The first one is to manage the synchronization requests from the receiver. It is also used to monitor the transceiver and inform when the transceiver has been successfully initialized – meaning mainly that the internal PLL is locked – and generally, it monitors the behavior of the global transmitter system.

1.2 Receiver (RX)

As well as the transmitter, the receiver is composed of the same 2 parts connected in the inversed order.

A high level block diagram of the reception is shown below:

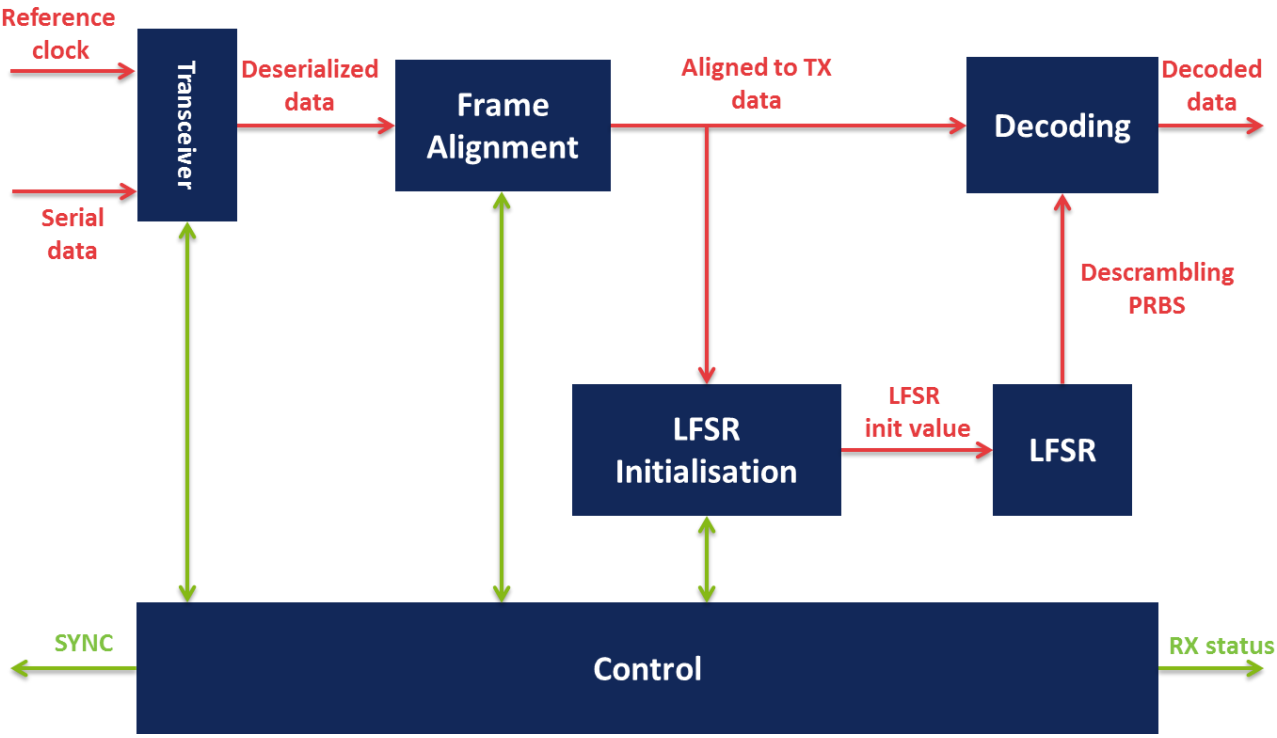

Figure 2: RX bloc diagram

The data coming from the transmitter goes first through the high-speed transceiver. It takes care of the deserialization of the flow and generally offers some equalization possibilities to balance the frequency response of the transmission channel. It requires a slow reference clock. This clock is used alongside the high speed data input to recover the clock from the data stream – this is done by the CDR (Clock and Data Recovery) prior to the deserialization. The high speed transceiver provides, then, the deserialized data at a slower speed as well as a synchronous clock.

The deserialized data are then fed to the frame alignment component that contains a small buffer – 3 frames thus a total of 48bits – and is used to align the frame in the receiver with the frame sent from the transmitter.

In normal operating mode – after synchronization – the frame are then processed to decode the ESIstream; this means that the disparity is processed and then the data are descrambled using the pseudo-random binary sequence provided by the LFSR.

During the synchronization, after the frame synchronization, the frame will be used to determine the initial value that should be provided to the LFSR to synchronize the scrambling and descrambling LFSR.

Comparable to the transmitter control, the receiver control is used to manage the synchronization. This implies that it generates synchronization requests when needed – at start-up and if the synchronization is lost. It also configures the buffer size for frame alignment and the LFSR initial value while the interface is synchronizing. Finally, it monitors the behavior of the global receiver system.