Application Note 41

MICRF500/501 RadioWire™ Clock Recovery Technique

By Per Kristian Bergseth; Bluechip Communication AS

Introduction

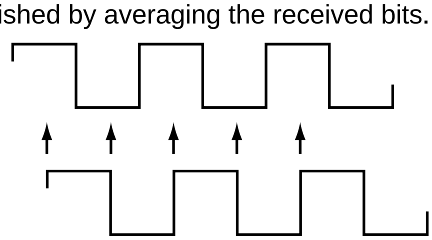
In Receive Mode, the MICRF50x RadioWire™ transceiver chip will continuously give an output. This output can be a valid data stream from a transmitter, or it can be random noise (no transmitter is sending data, or the link is out of range).

The output of the MICRF50x RadioWire™ transceiver (Data IXO pin) is the only pin that has data information. That is, no clock is provided from another pin.

Moreover, the transmitter and receiver frequencies might be slightly different from each other, due to VCO changes, different calibration in the transceiver pair, aging, jitter, etc. Thus, the received bits may be shorter or longer than a perfect bit, that is, the one that was transmitted. The transition from 0 to 1 (low to high) or vice-versa may be delayed. A 50% delay means that one bit is made 50% longer, and the following bit made 50% shorter.

Receiver Operation

To correctly read bits, it is necessary for the receiver to generate a sample clock. This clock should have the same period as the transmitter clock. Therefore, the term "regenerate clock" is also applied here. The clock should have a timeout (reach the end of one time period) in the middle of every bit period. In this case, a delay close to 50% can be tolerated. The transitions in the received signal is then expected to happen in the middle of the sample clock. This is illustrated in the figure below. It shows a perfect "101010..." data pattern being received. The recovered sample clock has the same period, and at every timeout, the input stream is sampled, that is, the sample clock has timeouts (reaches the end of a bit period) in the middle of the received signal elements (bit period), the transitions in the input stream occurs when the sample clock has reached half the period time, and the recovered clock has to be immune to noise. This can be accomplished by averaging the received bits.



Clock Recovery Implementation

The clock recovery procedure is based on transitions in the received signal. Transitions are expected when the Timer (in the microcontroller) has reached half of the bit period time. If they happen sooner or later, the difference is stored, and the Timer is adjusted based on the difference. Every difference will contribute to 1/16 to the adjusting value. A division is performed, and the remainder is stored and used for the next division.

Whenever a transition happens, the Timer Value is updated and stored whereby the transition time is captured and stored in the "Edge_time" register and the Remainder of the division is stored in the "Remainder" register.

This is the procedure before starting:

- Registers are filled with the ideal conditions: Edge_time = 1/2 period and Remainder = 0.
- Timer is initiated to have a period (p) = length of bit period (Timer increments until p is reached and then rolled over to 0).
- Timer roll-over interrupt is enabled.
- Timer is not running.

Starting the procedure:

- Search for a transition 0 to 1 or 1 to 0 in the received signal.
- When a transition is found, start the Timer.

During the procedure:

- Continuously search for transitions.
- If a transition happens, capture the value in the Timer and search for an opposite transition.

Interrupt:

- Sample and store the input.
- Test if a transition is captured.
- If no transition is captured since last interrupt, return from interrupt.
- If a transition is captured since the last interrupt:
  - Add Edge_time register and Remainder register.
  - Divide result by 16. Store result in the Edge_time register, store remainder in the Remainder register.
  - Compare Edge_time with the expected value. The expected value is (period/2)/16.
  - If Edge_time > expected value, the captured edge happened after the expected. Adjust (add) Timer with the difference Edge_time-expected.

MICREL, INC. 1849 FORTUNE DRIVE SAN JOSE, CA 95131 USA
TEL + 1 (408) 944-0800 FAX + 1 (408) 944-0970 WEB http://www.micrel.com