Application Note AN2358

Manchester Decoder Using PSoC®

Author: Philippe Larcher
Associated Project: Yes
Associated Part Family: All
Software Version: PSoC Designer 4.2
Associated Application Notes: AN2281, AN2325

Abstract
A Manchester Decoder can be built with two PSoC device digital blocks and some combinatorial logic. Once initialized, the decoder requires no firmware intervention. Clock and serial data recovered by the receiver can serve as inputs for a number of serial data communications methods including SPI and pattern recognition circuits.

Introduction
Manchester code is widely used in communications systems for reason of simplicity: a single signal conveys data and clock information, without the need for high-level protocol. Additional benefits include self-synchronization, zero DC components and independence from transmission media. A Manchester link consists of a transmitter (Manchester encoder) and a receiver (Manchester decoder).

Manchester encoding implementations have been described in previous Application Notes, particularly AN2281, "Manchester Encoder Using PSoC."

Manchester decoding is more complex, since it requires extracting clock and data information from a single signal. Application Note AN2325, "Serial Bit Receiver with Hardware Manchester Decoder" describes a method based on analog reconstruction of the clock signal. In contrast, this Application Note describes a Manchester Decoder based on digital reconstruction; implemented with only two PSoC digital blocks, it is portable to any PSoC sub-family.

Decoding speed is programmable and can easily achieve 200 kbps or more.

Manchester Code Principle
Manchester code embeds clock information with data in a very simple way: each bit is transmitted with a transition in the middle of bit time. For a '0', transition is 0 to 1, for a '1', transition is 1 to 0 (Figure 1).

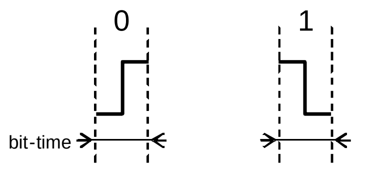

Figure 1. Manchester Coding for Bit Values 0 and 1

When transmitting successive bits, additional transitions are inserted between bits to satisfy the mid-bit transition rule, as represented in Figure 2.

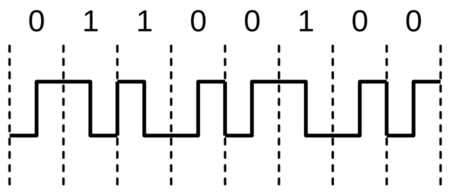

Figure 2. Transmitting Multiple Bits

Manchester Decoder, The Digital Way

The method described hereafter is not so much based on the direction of mid-bit edge, but on the fact that the bit value is present during the first half of bit time, before the transition edge. If a delay of three-fourths bit time is triggered by the incoming mid-bit transition, the value captured at the end of the delay will tell the next bit value (Figure 3).

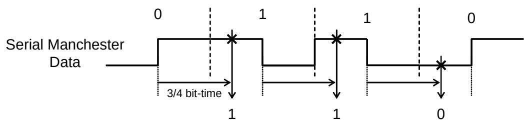

Figure 3. Capturing the Next Bit Value

It is important to note that if the next bit value is '1', the receiver sets a signal to invert the input bit stream polarity, so the next signal transition appears as a low-to-high transition (Figure 4). If the next bit value is '0', the receiver resets the inverted signal.

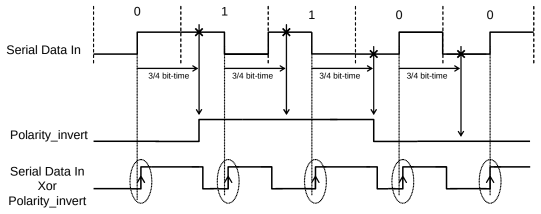

Figure 4. Inverting Mid-Bit Transition

I can hear you saying, "Okay, this is fine, but what's next?" Well, now simply rename "Polarity_invert" to "Serial Data Out" and "Serial Data In xor Polarity_invert" to "Serial Clock Out" and your Manchester receiver is done (Figure 5)!

Some considerations are still required for the first transition versus the idle state of Serial Data In, and will be treated at the end of this Application Note, after the description of implementation.

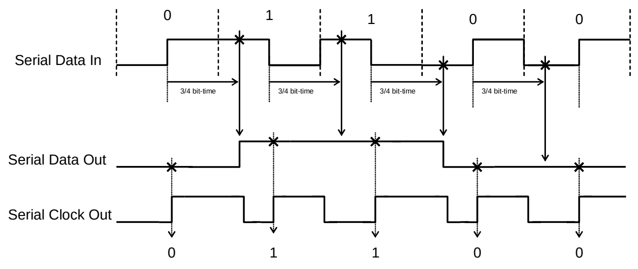

Figure 5. Manchester Receiver Outputs

PSoC Implementation

The previous description translates into the following block diagram.

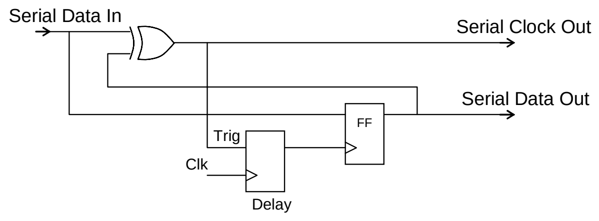

Figure 6. Manchester Receiver Block Diagram

As can be seen from the block diagram, the following three functions are required:

- A XOR gate to generate the Serial Clock Out.
- A D Flip-Flop to register the state of Serial Data In when the delay expires.
- A counter to generate a three-fourths bit time delay, triggered by the XOR-gate output (always a 0-to-1 transition).

Let's have a look at the PSoC implementation for these functions.

XOR Gate

This function is easily implemented with one of the look-up tables (LUT) placed on output rows, and does not require further explanation.

D Flip-Flop

There is no D Flip-Flop function directly available in the PSoC architecture. However, it is possible to implement a "conditional T Flip-Flop" with a digital block configured as a counter. If the counter period is set to '1' and the compare value to 'less than or equal 0', then the compare out signal will toggle upon each clock cycle when enable is high.

With this arrangement we just need to connect the enable signal to the XOR gate output, and the Polarity_invert signal only toggles when there is a change (0 to 1 or 1 to 0) in the input bit value, as seen in Figure 7.

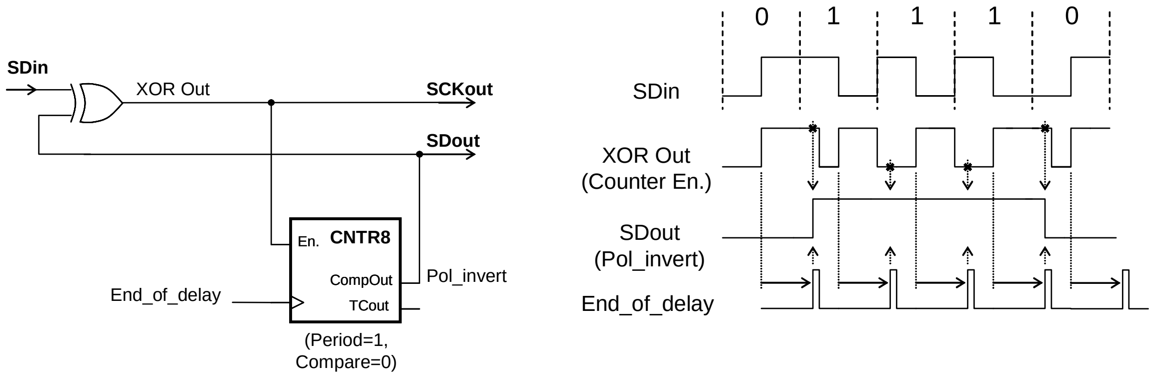

Figure 7. Emulation of D Flip-Flop with Counter and Enable

Delay Counter
The XOR gate output can be used as an enable signal to start counting. However, its duration may be only one-half bit time wide, which is insufficient to cover a three-fourths bit time counting period.

The problem is solved by OR-ing the initial enable signal with a counter compare out signal, thus extending the enable duration over the whole delay period, as shown in Figure 8. The combinatorial OR gate is implemented with a row LUT.

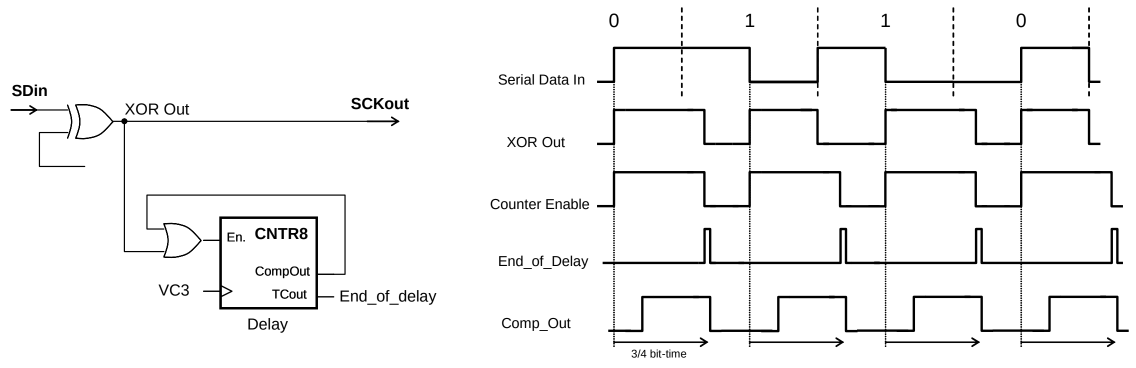

Figure 8. Three-Fourths Bit Time Delay Generation

Note that the polarity control is updated on the rising edge of TCout, causing the XOR out enable signal to be reset one clock cycle before CompOut, thus obviating any edge crossing.

VC3 (counter clock frequency) counts three fourths of bit time; this requires a VC3 clock at least four times the bit rate. However, tolerance must be added to cope with intrinsic precision and jitter of the transmitter and receiver.

Selecting a x16 oversampling rate gives more than 10% frequency tolerance on each side and is the retained value for the design.

For example, if the transmission bit rate is 100 kbps, then the clock frequency for the delay counter is 1.6 MHz, i.e. VC3 = SysClk/15.

Figure 9 shows the adequate value of period and compare out for a three-fourths bit time delay. These values depend only on the selected oversampling rate and do not need to be changed when changing the transmission speed through VC3.

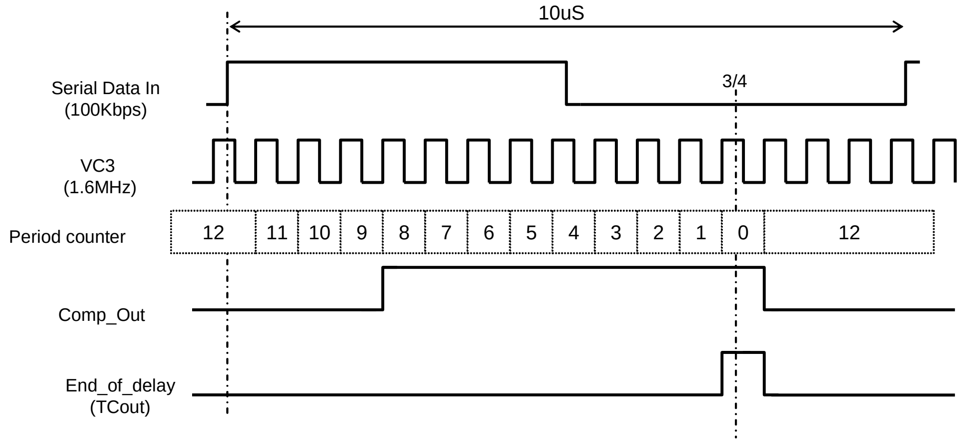

Figure 9. Period and Compare Out Values for Delay Counter

Place and Route

Figure 10a represents the whole Manchester receiver function. The PSoC device offers several possibilities to internally route back block output signals to block input (as required, for example, for XOR Out). However, internal feedbacks may create placement or pinout constraints if the receiver is imported into an existing design. For this reason, external feedbacks are recommended, at the cost of three additional input pins, as shown in Figure 10b. Figure 10c is a screen shot of the user module placement and routing in the associated PSoC Designer™ project.

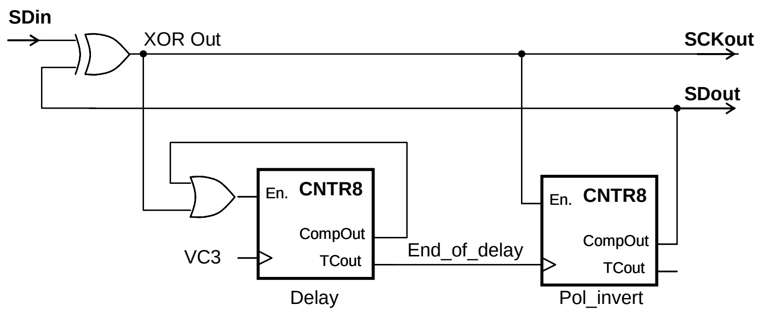

Figure 10a. Manchester Receiver Final

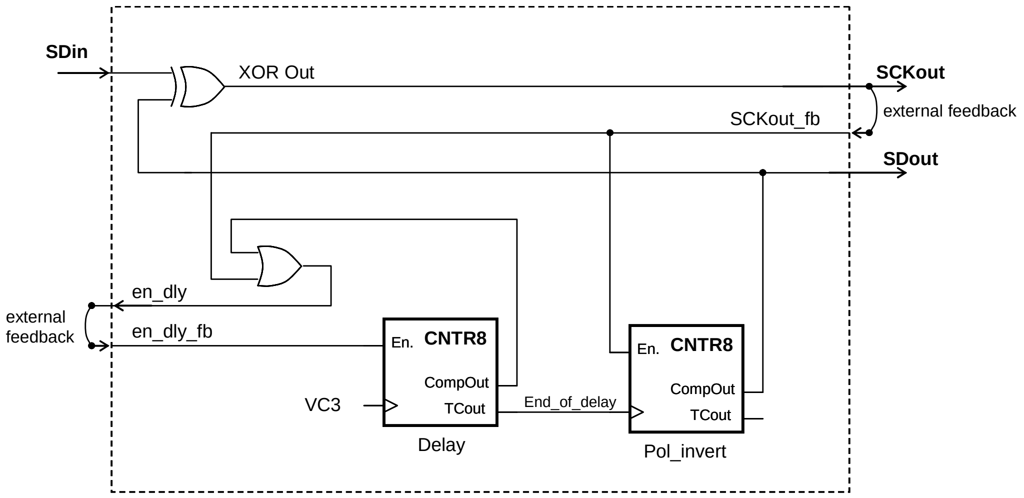

Figure 10b. Manchester Receiver for Flexible Place and Route

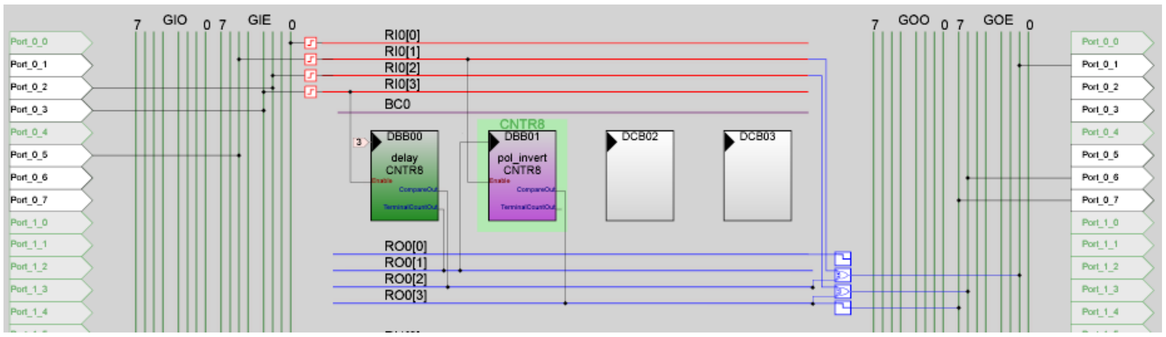

Figure 10c. Manchester Receiver, Device Editor Interconnect View

Porting the Manchester Decoder

The project associated with this Application Note can be used standalone or plugged into a larger application. The setup is simple.

Hardware

- Externally connect SCKout and SCKout_fb, en_dly and en_dly_fb.
- Set VC3 frequency to 16 times the serial bit rate.

Firmware Initialization

- Start the Delay counter and the Pol_invert counter.
- Re-adjust the Pol_invert counter period to ‘1’ (correct startup requires initial period to be ‘0’).

Synchronizing on the Right Edge

Up until now, the Manchester receiver behavior analysis assumed that the first detected edge was at mid-bit transition. However, there are cases where the first transition seen by the receiver is an inter-bit transition. For example, this occurs if the idle state of the line is low and the first received bit is a ‘1’ (or inversely), or, if the receiver is randomly enabled in the middle of a packet transmission. Such initial conditions translate into synchronization aliasing. However, a worthy receiver should be able to dynamically recover from this situation without manual intervention.

Initial State and First Bit Mismatch

Figure 11 shows a situation where the idle state of the serial input is low, and the first bit sent by the transmitter is a ‘1’. In good faith, the receiver will consider the first transition to be a mid-bit transition, and start extracting the serial data from this point.

Figure 11 shows that the faulty synchronization will cease as soon as a ‘0’ is received. This will cause the receiver to re-synchronize correctly and definitely, with the reason being that a ‘1-then-0’ bit sequence (and ‘0-then-1’) only exhibit mid-bit transitions and no inter-bit transitions, forcing receiver re-alignment on the right edge.

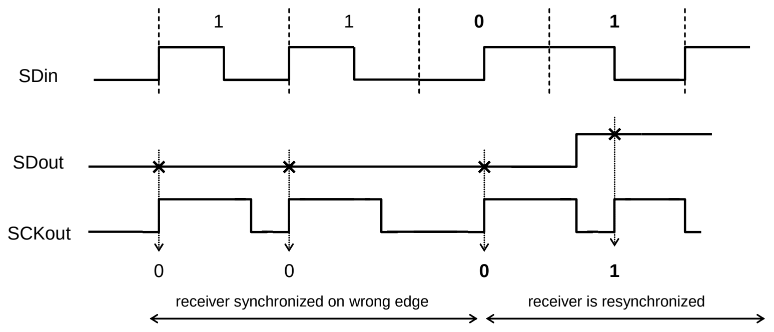
1
1
0
1
SDin
SDout
SCKout
0
0
0
1
receiver synchronized on wrong edge
receiver is resynchronized

Figure 11. Idle State Low and First Received Bit is ‘1’

The same story happens when the idle state of the line is high and the first received bit is a ‘0’, see Figure 12. In this case, the correct synchronization will start with the first ‘1’ that is received.

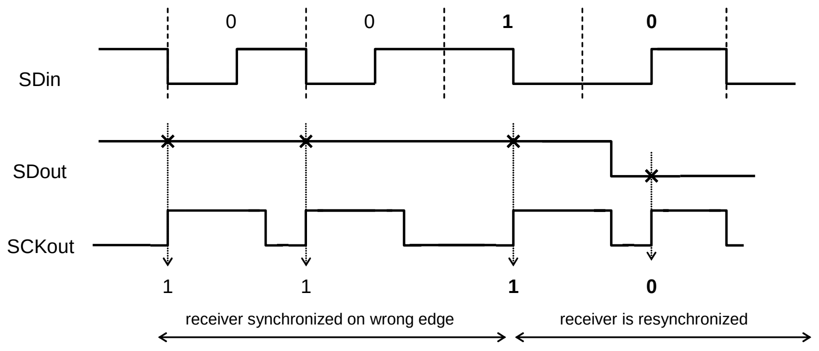
0
0
1
0
SDin
SDout
SCKout
1
1
1
0
receiver synchronized on wrong edge
receiver is resynchronized

Figure 12. Idle State High and First Received Bit is ‘0’

Synchronization errors due to hazardous enabling of the receiver during packet transmission create the same situations as described above, and are auto-recovered in the same way.

Auto-Synchronization

As can be seen above, the condition for correct synchronization is simple, and is up to the transmitter as follows:

- Either make sure that the first bit sent is compliant with the idle state of the line, e.g., always start messages with a ‘0’ bit when the idle state is low.
- Or, send a 0 to 1 (or 1 to 0) preamble to guarantee a correct synchronization in any situation. Of course, the preamble can extend beyond these two bits to add other benefits such as speed synchronization, pattern recognition, etc.

Conclusion

A robust Manchester Decoder has been implemented using two PSoC digital blocks. Recovered serial clock and serial data can feed a number of serial data communications methods including SPI and pattern recognition circuits.

About the Author

Name:	Philippe Larcher
Title:	Field Application Engineer, Cypress France
Background:	25 years of activity in computer and electronic system design. Authored several application notes, articles, and the book "VHDL, Introduction à la Synthèse Logique."
Contact:	ppl@cypress.com