



# Application Note

# AN2325

## *Serial Bit Receiver with Hardware Manchester Decoder*

**Author:** Volodymyr Sokil

**Associated Project:** Yes

**Associated Part Family:** CY8C29xxx

**PSoC Designer Version:** 4.2

**Associated Application Notes:** AN2091, AN2249, AN2281, AN2336

### Abstract

This Application Note describes implementation of a serial bit receiver. This receiver consists of a Manchester decoder implemented in hardware and an automatic preamble detection unit. The developed receiver may be used as a component in other communication systems.

### Introduction

Many different digital data encoding formats are used in communication systems. Usually, a data bit value is encoded by a signal's level, for example "1" – Vcc level, "0" – GND level. But in cases with special requirements, other encoding techniques must be used. One such popular scheme is Manchester encoding, in which data bits are represented by transitions from one logical state to another. Thus, a Manchester encoded signal does not contain a DC level. This balances the line drive and simplifies receiver data implementation.

When Manchester code is used, the length of each data bit is known and constant. The value of a bit is determined according to the direction of the transition. The transition from low to high represents a logical 0, and the transition from high to low represents a logical 1. Each bit contains a signal transition from one state to the next. This makes a signal self-clocking.

The main advantage of Manchester encoding is the fact that a signal synchronizes itself. This minimizes the error rate and optimizes reliability. More detailed information about encoding with an example of encoder implementation can be found in AN2281 "Manchester Encoder Using PSoC™."

There are many different methods for decoding Manchester encoded data. One of them is described ahead.

### Serial Bit Receiver Flowchart

As can be seen from AN2281, a data can be decoded using the exclusive or operation (XOR) with a clock signal. So it is necessary to restore a clock from the encoded data.

Manchester encoding is a form of clock phase manipulation. Clock restoration requires the elimination of signal phase shifting. For this purpose, we may use the method of multiplying the data signal by 2. After this operation, the clock's second harmonic signal is obtained. A resonant system with a tuned frequency equal to twice the clock frequency is used to reconstruct the omitted signal pulses.

Let's try to use this concept to implement a hardware serial bit receiver. The receiver flowchart is shown in Figure 1. Annotations from 1 to 7 define signal checkpoints. Signal waveforms are shown in Figure 2.

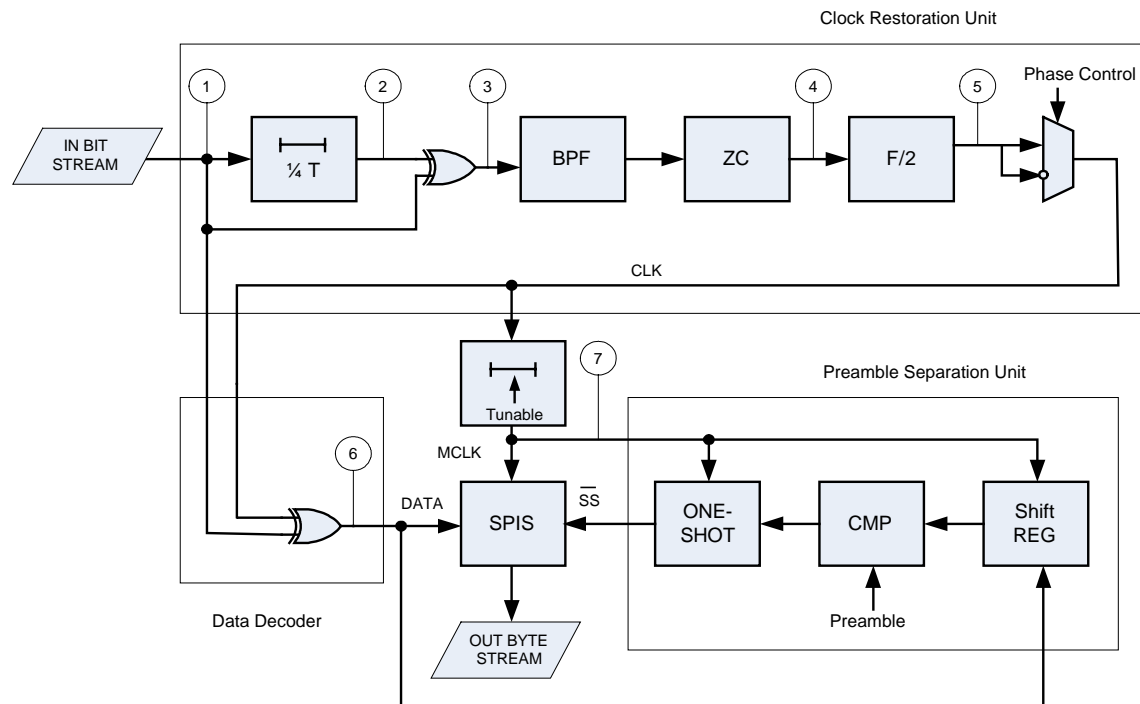


Figure 1. Serial Bit Receiver Flowchart

The input data signal is shown on waveform 1 in Figure 2. For the purpose of multiplication, the input signal is shifted by  $\frac{1}{4}$  of the clock period (waveform 2) and XORed with the original signal. This signal shifting generates a double frequency signal with a 50% duty cycle, but with some gaps. And, accordingly, this signal has maximum first harmonic amplitude at the doubled frequency. The resulting signal is shown in waveform 3. As the resonant system for signal reconstruction, a band-pass filter (BPF) is tuned to the doubled clock frequency. A zero-crossing detection circuit on the BPF output generates the digital signal (waveform 4). The frequency of the obtained signal is divided by 2 – and the data clock signal (waveform 5) and, accordingly, the data signal (waveform 6), is reconstructed! The SPI receiver in slave mode forms the byte stream.

But, the data clock signal reconstruction circuit (mostly BPF) inserts some phase shift into the restored data clock signal. So the decoded data signal is obtained with a small amount of phase error. For correct SPIS operation, the reconstructed clock (before use as the MCLK signal) is delayed. The delay time is tuned to obtain the SPIS sample moments at the middle of the data bit.

For synchronization on byte level, an input bitstream must contain a synchronization bit sequence. In this example, the two-byte synchronoword 0xFF 0x0F is used. The first byte of this sequence (0xFF) is recognized by the SYNC module, which controls SPI receiver operation using the slave select input signal. The second byte (dummy) is omitted by the SPI receiver for byte synchronization. Detailed information about preamble detection and extraction and implementation can be found in AN2336 “Hardware Bitstream Sequence Recognizer.”

Note that the initial state of the frequency divider can be undefined. So the data clock can be reconstructed in inverted form (waveform 5'). In this case, the data would also be decoded in inverted form (waveform 6'). To recognize this situation, the encoded bit sequence must not contain the inverted first byte of the synchronoword (0x00). Therefore, the absence of synchronization for one second, for example, indicates that the repaired clock signal must be inverted by switching the multiplexer. The phase control signal is formed by the preamble separation unit.



Note that since the BPF center frequency is equal to 200 kHz, the Op-Amp Bias global resource and BPF power must be set to High.

The next stage is the data signal decoding. The restored data clock signal from the PWMOutput of the CLK\_DELAY module is XORed by the Row\_2\_Output\_0 LUT function with the input data signal. Decoded data are fed to the SPIS, placed in DBB31. The restored clock signal is delayed by the PWMDB8 CLK\_DELAY module, placed in DBB21 and DBB21, and used as the MCLK. Setting of delay time is provided by the DBCounter clock routed to the SysClk 24 MHz using the CLK\_DELAY\_DB\_OUTPUT\_REG direct write. Detailed information about using the PWMDB User Module as a delay element can be found in its data sheet in PSoC Designer™. The oscilloscope waveforms for these stages of decoding are shown in Figures 5 and 6 of Appendix A.

For byte synchronization, the clock and data signals are used simultaneously by the synchronization unit. The PRS, SYNC\_EXTR, is placed in DCB33 and used as the synchroword recognizer. The CompareOut output is fed to the one-shot user module, SPIS\_SS, placed in DBB31, in one-shot mode. This module must delay the SYNC\_EXTR signal for 6½ periods. For this it uses the inverted data clock signal from buffer BUF1, placed in DBB30. The delayed signal is used as the slave select input signal (~SS) for the SPIS User Module.

After power-up, it is possible that different undefined data can be received during the start-up period. To prevent synchronization errors, the synchroword recognizer is initially blocked. When the start-up period (in this project, 15 ms) has elapsed, the Sleep\_Timer ISR unblocks the synchroword recognizer by setting the INIT\_BIT in the SYNC\_FLAG variable.

When the PML\_EXTR Shift register value is equal to the Seed register (first byte of the synchronization word was received by SPIS) the CompareOut signal goes high for one data clock period. The signal, delayed by SPIS\_SS, resets the SPIS on the next byte boundary. Moreover, the SPIS\_SS interrupt handler sets the synchronization bit, SYNC\_BIT in SYNC\_FLAG. The Sleep\_Timer interrupt handler checks SYNC\_FLAG after a one second time period. If the SYNC\_BIT is not set, the repaired clock signal is inverted using the RDI2LT0 and RDI2LT0 registers.

## Conclusion

This Application Note describes hardware implementation of a serial receiver with a Manchester decoder and automatic preamble synchronization. A software implementation of the Manchester decoder can be applied for low (tens) kbits/s. The main advantage of the developed receiver is that hardware implementation can be easily tuned for different input bit rates, up to hundreds kbits/s.

Two example projects were developed for testing the receiver. The first is tuned for a data bit rate equal to 100 kbit/s with 10% tolerance. The second is tuned for a data bit rate equal to 10 kbit/s. In both cases, the receiver demonstrates excellent operating stability and resynchronization in the case of lost synchronization.

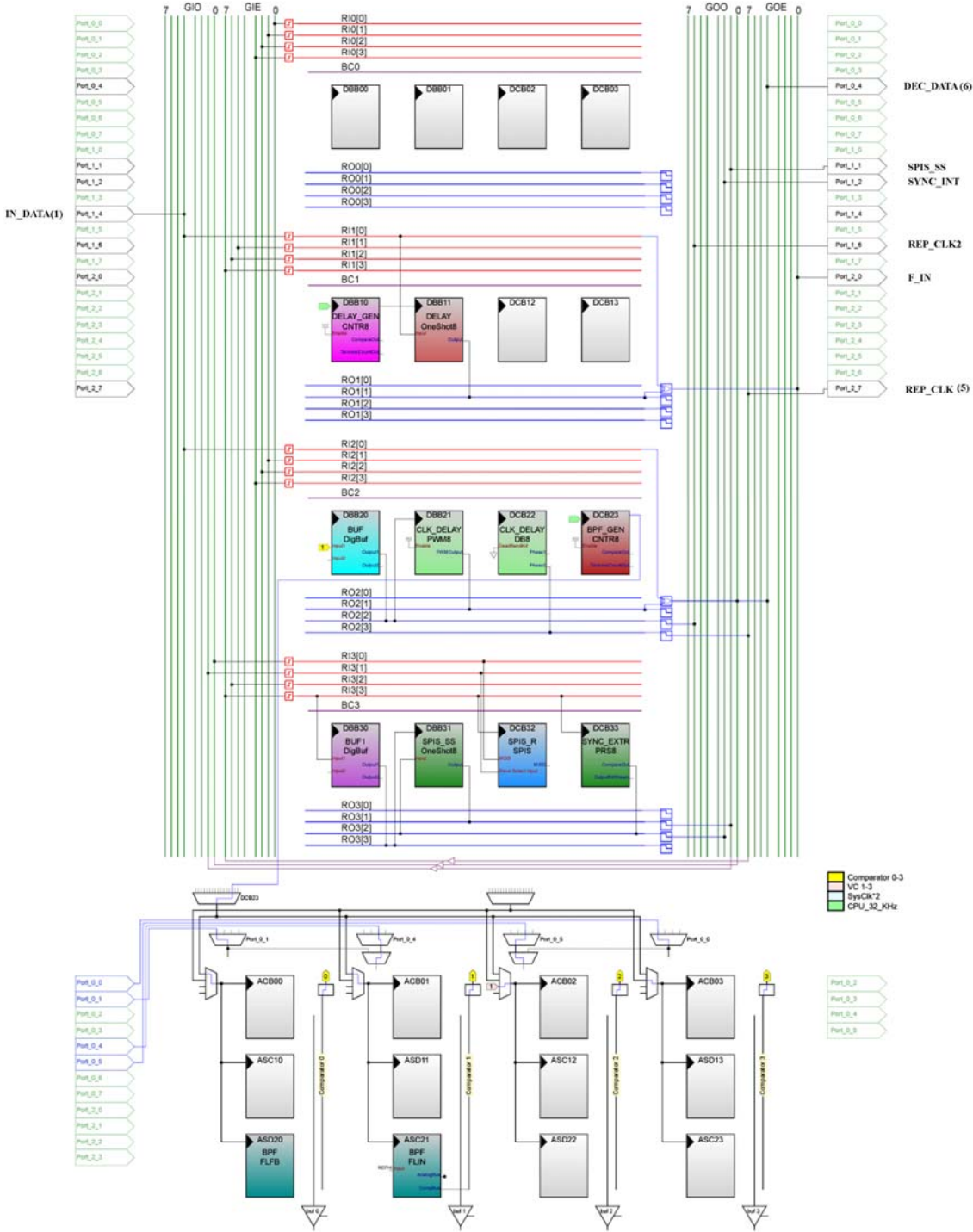
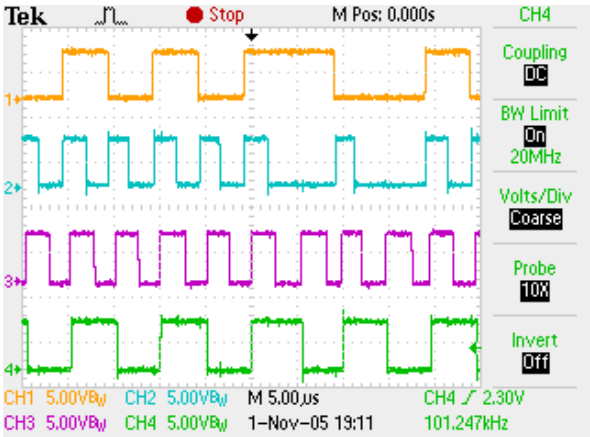


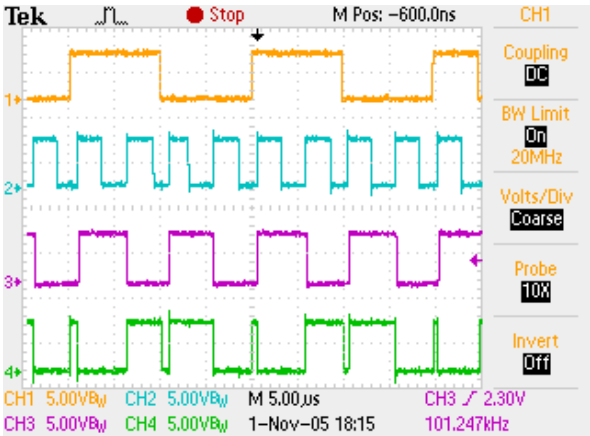
Figure 3. PSoc Receiver Internal User Module Configuration

### Appendix A



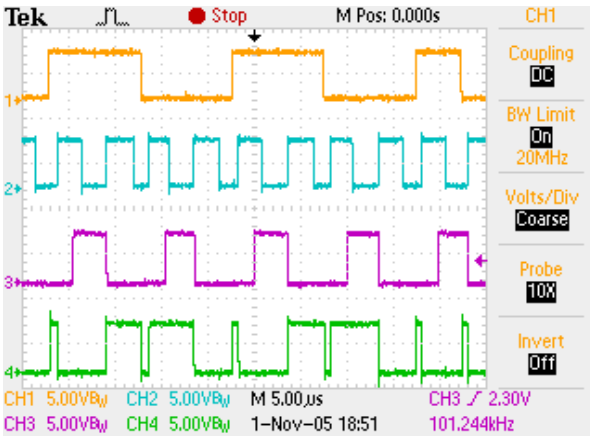
- CH1 – IN DATA
- CH2 – IN BPF
- CH3 – OUT BPF
- CH4 – RECONSTRUCTED CLK

Figure 4. Restored Data Clock Signal Waveform



- CH1 – IN DATA
- CH2 – RECONSTRUCTED CLK\*2
- CH3 – RECONSTRUCTED CLK
- CH4 – DECODED DATA

Figure 5. Decoded Data Waveform



- CH1 – MCLK
- CH2 – RECONSTRUCTED CLK\*2
- CH3 – MCLK
- CH4 – DECODED DATA

Figure 6. Received Data Waveform

---

## About the Author

**Name:** Volodymyr Sokil

**Title:** Post-Graduate Student

**Background:** Volodymyr earned a diploma in computer engineering in 2001 from National University "Lvivska Polytechnika" (Lviv, Ukraine), and is currently a post-graduate student at this University. His interests involve embedded systems design and information security.

**Contact:** [sokilm@ukr.net](mailto:sokilm@ukr.net)

---

Cypress Semiconductor  
2700 162<sup>nd</sup> Street SW, Building D  
Lynnwood, WA 98087  
Phone: 800.669.0557  
Fax: 425.787.4641

<http://www.cypress.com/>

Copyright © 2005 Cypress Semiconductor Corporation. All rights reserved.

"Programmable System-on-Chip," PSoC, PSoC Designer and PSoC Express are trademarks of Cypress Semiconductor Corp.

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

The information contained herein is subject to change without notice. Made in the U.S.A.