ANALOG DEVICES

AN-915
APPLICATION NOTE

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

CDR Operation for ADF7020, ADF7020-1, ADF7021, and ADF7025

by Austin Harney and Philip Quinlan

INTRODUCTION

The clock and data recovery (CDR) module in the ADF7020, ADF7020-1, ADF7021, and ADF7025 devices is implemented using an oversampled digital phased-locked loop (DPLL), operating at 32× the transmit data rate. The CDR PLL resynchronizes the received bit stream to a local bit clock, Rx clock.

The phase detector in the DPLL measures the phase error by comparing the time between bit transitions in the recovered bit stream and the rising edge of a local bit clock. A numerically controlled oscillator (NCO) generates the recovered clock. When a bit transition is detected at the output of the post-demodulator, the phase of the NCO output is adjusted by +1/32, 0, or −1/32 of a bit time.

A simplified block diagram of the CDR is shown in Figure 1.

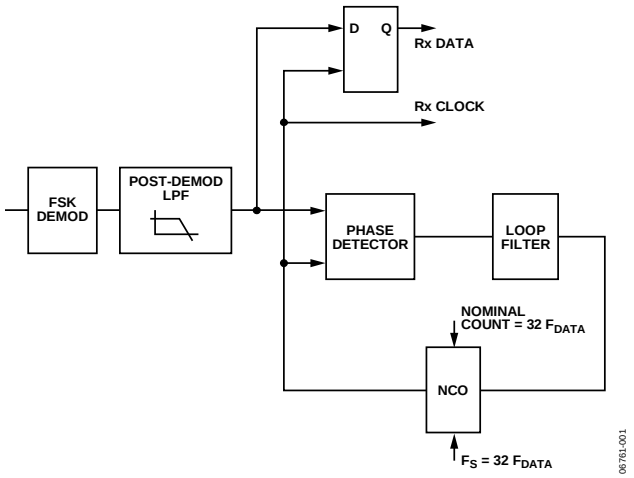

Figure 1. Block Diagram of Symbol Timing Recovery PLL

MAXIMIZING DATA RATE TOLERANCE

The data rate tolerance of the CDR is dependent on the number of bit transitions in the transmit bit stream. This tolerance is maximized, and the CDR lock time is minimized when a maximum transition bit pattern, 10101010…, is used as a preamble. A maximum NCO phase adjustment of 1/32 of a bit period is permitted in a single bit period. This results in a maximum data rate tolerance of 1/32 × 100 = ±3.13% with a 101010… preamble.

However, this data rate tolerance is reduced in the data field, as shown in Figure 2, where bit transitions are not guaranteed to occur at regular intervals. In general, it is the run-length limit (RLL) properties of the transmit data field that determine the actual data rate tolerance of the CDR.

The RLL property of a code defines the maximum number of identical contiguous bits in the encoded bit stream. In general, all encoding schemes are defined by a (d, k) constraint, where d, k represents the minimum and maximum number of identical symbols between unequal symbols.

As an example, a code with a (d, k) equal to (0, 4), has no more than four contiguous identical bits. Thus, in general, it is the k constraint of the code used for data field encoding that determines the maximum data rate tolerance.

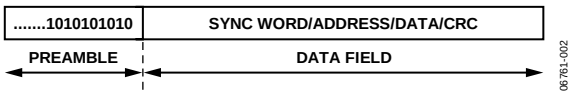

Figure 2. Typical Message Format

As an example, Manchester encoding has a maximum RLL of 2 bits, that is, the maximum number of contiguous bits in the encoded Manchester sequence is 2 bits. For example, a Manchester encoder input code word of 0101 results in an output code word of 01100110 at twice the input data rate. Thus, the maximum tolerance is ±3.125%/2 = ±1.56% of the Manchester encoded output rate. In practice, however, the tolerance is larger than this because the average RLL at the Manchester encoder output is less than 2. Simulations show that, given a random input binary bit sequence, it is possible to support a data rate tolerance of approximately ±2% using Manchester encoding.

Another low complexity encoding scheme that can be used to maximize data rate tolerance is to insert additional data bits at specific time intervals in the transmit bit stream to guarantee a specific maximum RLL. This is referred to as a bit stuffing code.

The advantage to this scheme is that it is simple to implement, and it does not suffer from the high code rate loss that exists with Manchester encoding, where the code rate is 1/2.

TABLE OF CONTENTS

Table 1 summarizes the data rate tolerance of these encoding schemes. Other higher rate codes, such as an 8/9 rate code or an 8/10 rate code, can be considered at a cost of increased hardware or software complexity.

Table 1. DPLL Frequency Tracking Range

Data Encoding Scheme	Tolerance
1010101... Preamble	±3.125%
Manchester Code, Max RLL = 2, Random Input Data	±2.00%
Bit Stuffing Code with RLL = 3, Random Input Data	±1.60%

MAXIMIZING RUN LENGTH CONSTRAINTS

In applications such as wireless streaming, or when large packet-based protocols are used, some form of data encoding is required that guarantees a finite RLL constraint in the data field to support proper CDR operation.

However, it is possible to tolerate uncoded data fields or to support data fields with large RLL coding constraints if the data rate tolerance and packet size are minimized. This is normally the case in packet-based wireless sensor telemetry applications where the data payload is typically less than 64 bytes. In addition, the data rate tolerance is typically tightly controlled because it is set by matching the transmit and receiver crystals. This typically results in better than ±50 ppm or 0.005% nominal frequency error if the data rate is chosen to be an exact submultiple of the crystal.

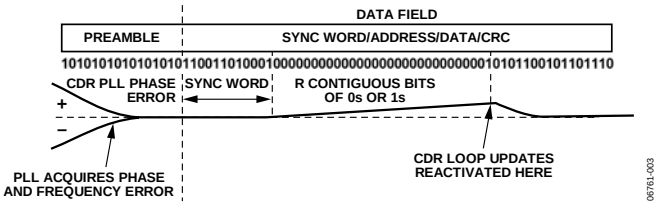

Figure 3. Phase Error Trajectory for Uncoded Data Field

An example is shown in Figure 3. A 101011010… preamble is used to achieve fast bit synchronization and to track frequency errors due to data rate differences between Tx and Rx. At the end of the preamble, the CDR continues to track the phase and frequency errors while the sync word is recovered. The long sequence of zeros that follows the sync word means that the CDR does not see any symbol transitions to permit tracking of frequency errors. During this period, the CDR phase error monotonically increases (or decreases, depending on the sign of the frequency error) until further data transitions occur in the transmit bit stream.

In this scenario, the CDR can still recover data provided the maximum phase error stays within approximately ±10% of a bit period. The rate of increase of the CDR phase error is proportional to the data rate tolerance, and minimizing this is important if long RLL values must be supported. Thus, large RLL constraints can be accommodated if the data rate tolerance and/or packet size are minimized to restrict the PLL phase accumulation in such conditions.

A relationship between data rate tolerance and RLL constraints can be established from the following:

$$Phase = 2\pi\, Ft$$

$$\frac{\Delta\Phi}{360} = \Delta F\, \Delta t$$

where Δt is defined in terms of the number of contiguous data bits, R.

If the phase accumulation is restricted to 1/10 of a bit period

$$0.1 = \Delta F_{DATA}\, R\, Tbit$$

where $Tbit = 1/F_{DATA}$

then

$$0.1 = \left(\frac{\Delta F_{DATA}}{F_{DATA}}\right) R$$

For a given data rate tolerance ($\Delta F_{DATA}/F_{DATA}$), the number of identical contiguous data bits, R, that can be tolerated is

$$R = \left(\frac{F_{DATA}}{\Delta F_{DATA}}\right) 0.1$$

Table 2 lists examples of the constraint length R vs. data rate tolerance.

Table 2. Constraint Length R and Data Rate Tolerance

Data Rate Tolerance % ($\Delta F_{DATA}/F_{DATA}$)	$F_{DATA}/\Delta F$	R Bits
0.1%	1000	100
0.05%	2000	200
0.01%	10,000	1000

This analysis assumes that R contiguous data bits occur in isolated segments of the packet. Consider the case where such bit patterns are separated by short runs of random data bits that may not permit the CDR to fully recover the phase error in between the bursts of contiguous data. In such cases, it is best to assume that the RLL constraint is the total length of the combined bursts and to then compute the data tolerance requirements (that is, the Tx and Rx Xtal matching requirements) based on this increased R value.

For cases where the RLL is unknown, or in the case of uncoded data fields where the entire packet can have contiguous data bits, the data tolerance requirements should be based on an RLL value equal to the total packet length (R = total packet length).

In general, packet scrambling should be considered as a technique to break up and randomize long bit patterns of contiguous data.

SUMMARY

In streaming applications where tens of kB may be sent over the air, it is advisable to use some form of data encoding to minimize the RLL. This allows the CDR circuit to successfully track

the incoming data even in the presence of long strings of consecutive 1s or 0s.

For bursty packet-based systems, for example, where the payload is typically fewer than 64 bytes, the CDR can tolerate long consecutive strings of 1s or 0s if the nominal CDR frequency error between the CDR_CLK and 32 × DATARATE is minimized.

This is best done by choosing an appropriate DATARATE and crystal combination so that the DATARATE is an exact sub-multiple of the crystal. For example, to minimize the CDR frequency error for a 9.6 kbps DATARATE, an 11.0592 MHz crystal is a good choice. In that case, the nominal frequency error is simply the frequency error between the transmit and receiver crystals, which typically results in better than ±50 ppm or 0.005% nominal frequency error. This frequency error allows 2000 contiguous bits. Alternatively, a 12 MHz crystal yields a nominal frequency error of close to 2.2% and, therefore, a maximum RLL of only 2 bits.