Using PLLs to Obtain Carrier Synchronization: Part 1

Louis Litwin, Thomson Multimedia - March 25, 2003

Using PLLs to Obtain Carrier Synchronization: Part 1
An important task for a digital communications receiver is to remove any frequency/phase offsets that might exist between the transmitter and receiver oscillators. Designers typically implement a carrier-recovery loop to remove this offset. A key component of a carrier-recovery loop is a phase-locked loop (PLL). The use of a PLL enables the receiver to adaptively track and remove frequency/phase offsets.

In this two part series, we'll take a look at the carrier-recovery problem and examine how PLLs can be used to obtain carrier synchronization in a digital communication system. In Part 1, we'll look at the carrier-recovery problems designers face and provide the basics on PLL design. In Part 2, we'll further the discussion by looking at phase detector design and carrier recovery techniques.

Analyzing the Carrier Recovery Problem
Digital communications systems can be divided into two classifications depending on the frequency range used for signal transmission. The term baseband is used to describe a system where the transmitted signal spectrum is centered around DC (i.e., it is centered around 0 Hz). Baseband transmission is typically used in a wired environment. An example of a baseband system is a data modem that operates on telephone lines using the 0 to 4 kHz voiceband.

In contrast, the signals in a wireless communications system need to be transmitted at RF frequencies in order for them to propagate efficiently. The term passband is used to describe those systems whose transmitted signal spectrum is centered around a non-zero frequency that is typically much higher than the highest frequency component of the data signal.

The spectrums for a baseband and passband communications system are shown in Figure 1

Figure 1: Diagram of baseband and passband comm systems. In this diagram, the left plot shows the
spectrum of a baseband system and the right plot shows the spectrum of a passband system where the frequency of the carrier is $f_c$ Hz.

The data signal in a passband digital communications system is modulated onto a sinusoidal carrier with a frequency of $f_c$ Hz by multiplying the data and carrier signals together. This multiplication is a frequency conversion process that shifts, or upconverts, the data in frequency so that the spectrum is centered around the carrier frequency of $f_c$ Hz. The signal is then transmitted over the communications channel at this frequency.

The frequency-conversion process must be reversed at the receiver in order to bring the data signal back down to baseband. Ideally this step can be accomplished by multiplying the received data signal by a locally generated sinusoid whose frequency is also $f_c$ Hz and whose phase is identical to that of the carrier signal used at the transmitter.

From frequency mixing theory it is known that doing so will produce two images, or versions, of the data signal. One image will be centered at a frequency equal to the sum of the frequencies of the carrier signal and the locally generated sinusoid. Since these frequencies are ideally identical, this image will be located at a frequency of $2f_c$ Hz. The image at this high frequency is undesired and can be removed by a lowpass filter.

The other desired image is centered at a frequency equal to the difference of the frequencies of the carrier signal and the locally generated sinusoid. This desired image is centered at DC since the difference of the identical frequencies is 0 Hz. Thus the process of multiplying the received signal by a locally generated sinusoid and subsequently lowpass filtering the signal accomplishes the task of bringing the signal back down to baseband.

**Theory vs. Reality**

Of course in practice the situation is not so simple. The locally generated sinusoid will have both an unknown phase offset and an unknown frequency offset from the transmitter's carrier signal. These offsets are the result of real-world issues such as oscillator drift due to temperature and time. Ignoring the simpler problem of phase offset for a moment, let us assume that the frequency of the local oscillator at the receiver is:

\[
 f_{LO} = f_c + \Delta f
\]

where $\Delta f$ represents the amount of frequency offset present at the receiver's local oscillator. Multiplying the received signal by the local oscillator will produce the previously discussed sum and difference images. The sum image will be centered at a frequency of:

\[
 f_{sum} = f_{LO} + f_c = f_c + \Delta f + f_c = 2f_c + \Delta f
\]

This image can again be removed by a lowpass filter. Similarly, the desired difference image is centered at a frequency of:
Thus, in the presence of a local-oscillator frequency offset, the downconverted data signal is not at baseband. Instead, it is centered around the frequency $\Delta f \text{ Hz}$. The frequency $\Delta f$ is the amount of the difference between the transmitter's carrier frequency and the receiver's local-oscillator frequency. Since $\Delta f$ is typically much smaller than $f_c$, the resulting downconverted signal is said to be at near-baseband.

If we denote the transmitted signal as $x(t)$ and its corresponding spectrum as $X(f)$, then the near-baseband version of the signal in the frequency domain can be written as $X(f - \Delta f)$.

Recall from Fourier transform theory that the corresponding time domain near-baseband signal is thus $x(t)e^{j2\pi \Delta f t}$. This multiplication by a complex exponential in time will have the undesirable effect of causing the data signal’s constellation to circularly rotate, or spin, over time. This spinning constellation means that the receiver's symbol decision device, or slicer, cannot make accurate symbol decisions. Hence this frequency offset must be removed before further processing of the signal can occur.

Since the local oscillator (running at a frequency of $f_c + \Delta f \text{ Hz}$) is typically an analog component, the easiest way to remove the residual frequency offset of $\Delta f \text{ Hz}$ is to multiply the signal by another sinusoid with a frequency of $\Delta f \text{ Hz}$. The multiplier used to remove this offset is called a derotator because it derotates the signal back to baseband.

The task of generating a sinusoid with a frequency of $\Delta f \text{ Hz}$ is accomplished by the carrier-recovery loop. Since the carrier-recovery loop is essentially a specific use of a PLL, we will start by discussing the theory of phase-locked loops. Afterwards a practical method for carrier recovery will be introduced along with simulation results showing the performance of the algorithm.

**The Goal of the PLL**
The basic problem addressed by a PLL is how to generate a sinusoid that is phase-locked with an externally generated sinusoid of unknown (and possibly varying) frequency and phase. This problem can be stated mathematically as follows. Let $x(t)$ be an externally generated sinusoid where $x(t) = \cos(\theta_x(t))$. The term $\theta_x(t)$ is the angle of the sinusoid. It is expressed in radians and it can be written out as $\theta_x(t) = 2\pi f_x t + \Phi_x$. $x(t)$ is thus a sinusoid with a frequency of $f_x \text{ Hz}$ and a phase of $\Phi_x$ radians. To simplify the discussion that follows, we will assume that $\Phi_x = 0 \text{ radians}$ so that it can be removed from the expressions.

The PLL creates a locally generated sinusoid $y(t) = \cos(\theta_y(t))$ where $\theta_y(t) = 2\pi f_y t + \Phi_y$. The task of the PLL is to modify $\theta_y(t)$ until $f_y = f_x$ and $\Phi_y = \Phi_x$. When those conditions are met, the locally generated sinusoid will be phase-locked to the externally generated sinusoid.

**PLL Elements**
A PLL consists of three basic elements: a phase detector, a loop filter, and a numerically controlled oscillator (NCO). The block diagram in Figure 2 shows how these elements are connected in order to form a PLL.
The goal of this PLL is to produce a locally generated sinusoid $y(t)$ whose instantaneous angle is equal to the instantaneous angle of the received signal $x(t)$. The following paragraphs will briefly describe these three key elements.

1. **The Phase Detector**
   The purpose of the phase detector is to generate an error signal that drives the PLL. This error should be proportional to the phase difference between the signals $x(t)$ and $y(t)$.

   There are numerous ways in which the phase detector can be implemented. The type of phase detector used often depends on the application, such as whether the PLL is being used for a carrier recovery or timing recovery loop. The main trade-offs to consider when choosing a phase detector are implementation complexity and the "quality" of the error generated. The quality of the phase detector can be thought of in terms of how closely the phase detector's error approximates the error generated by an ideal phase detector (the ideal phase detector will be presented in the next section).

2. **The Loop Filter**
   The purpose of the loop filter is to filter the phase error signal $\varepsilon(t)$ in order to provide a better signal to the NCO. The error signal generated by the phase detector is actually a noisy estimate of the phase error, i.e. the error signal consists of an error term and a noise term.

   The loop filter processes the phase error signal $\varepsilon(t)$ in order to generate a useful error while suppressing the effect of the noise as much as possible. The values of the gain parameters chosen for the loop filter control the loop bandwidth of the PLL. The size of the loop bandwidth determines the range of error signal frequencies that the loop filter will pass.

   The value for this bandwidth has a direct impact on the performance of the PLL. If the value of the loop bandwidth is large, the loop filter can pass a wide range of frequencies for the error signal. A wide loop bandwidth will thus allow the PLL to track out large frequency errors. However, it will also pass a wider portion of the noise spectrum. The end result is a noisy control signal for the NCO and that translates into phase jitter on the locally generated signal $y(t)$.

   In contrast, a small value for the loop bandwidth will limit the amount of noise that passes through the filter. The narrow loop bandwidth will result in a cleaner control signal for the NCO. However, the drawback of using a narrow loop bandwidth is that the range of frequencies that the PLL can track out is reduced as well.
In addition to the loop bandwidth, another important design criterion for the loop filter is the order of the filter. A first-order loop filter simply multiplies the error signal by a proportional gain $K_p$. The output of this filter can be written mathematically as:

$$
\varepsilon_{\text{loop}}(t) = K_p \varepsilon(t)
$$

From control theory it is known that such a loop filter can be used to completely track out a phase error. This result can be illustrated mathematically as follows. In the presence of a phase offset of $\Phi$ radians, the steady-state error of the PLL (as derived in Digital Communication\textsuperscript{2}) is given by the following $z$-transform equation:

$$
\mathcal{E}_s(t) = \lim_{z \to 1} \frac{(z-1)z\Phi}{zL(z)+z-1}
$$

where $L(z)$ represents the $z$-transform of the loop filter. For a first-order loop filter, $L(z) = K_p$. Thus the steady-state error is:

$$
\mathcal{E}_s(t) = \lim_{z \to 1} \frac{(z-1)z\Phi}{K_p z + z - 1} = 0
$$

The interpretation of the above equation is that, in the presence of a phase offset, a PLL with a first-order loop filter will converge to zero error. This result can also be understood intuitively because the error signal $\varepsilon(t)$ represents the instantaneous phase offset and that offset is being used to drive the NCO.

However, a PLL with a first-order loop filter cannot track out a frequency offset. The steady-state error for a frequency offset of $f$ Hz was derived in Digital Communication\textsuperscript{2} to be:

$$
\mathcal{E}_s(t) = \lim_{z \to 1} \frac{z(2\pi f)}{zL(z)+z-1}
$$

Substituting in $L(z) = K_p$, the steady-state error for a frequency offset is found to be:

$$
\mathcal{E}_s(t) = \lim_{z \to 1} \frac{z(2\pi f)}{K_p z + z - 1} = \frac{2\pi f}{K_p}
$$

Thus, the first-order PLL will converge to a non-zero value in the presence of a frequency offset. In order for the PLL to also track out a frequency offset, a PLL with a second-order loop filter is needed.

A second-order loop filter consists of two paths (Figure 3). The proportional path multiplies the error signal by a proportional gain $K_p$. In addition, an integral path multiplies the error signal by an
integral gain $K_i$ and then integrates the scaled error using an adder and a delay block.

Figure 3: Structure of a typical second-order loop filter.

The output of the second-order loop filter can be expressed mathematically as:

$$\varepsilon_{\text{loop}}(t) = K_p \times \varepsilon(t) + K_i \times \int_0^t \varepsilon(i) \, dt$$

A PLL that implements such a filter is capable of tracking out both a phase and a frequency error.

We will apply the same mathematical equations from the first-order loop filter example in order to show that the second-order PLL can also completely track out a frequency error. The z-transform of the second-order loop filter is

$$L(z) = K_p \frac{z + K_i}{z - 1}$$

By substituting that expression into the equation for the steady-state error in the presence of a frequency offset, we obtain:

$$\varepsilon_{\text{ss}}(t) = \lim_{z \to 1} \frac{z(2\pi f)}{L(z) + z - 1}$$

$$= \lim_{z \to 1} \frac{(2\pi f)}{L(z)}$$

$$= \lim_{z \to 1} \frac{(2\pi f)}{K_p \frac{z + K_i}{z - 1}}$$

$$= \lim_{z \to 1} \frac{(z - 1)(2\pi f)}{K_p (z + K_i)} = 0$$

Thus, the second-order PLL is able to converge to a zero error even with a frequency offset. This result can be understood intuitively by recalling that the integral of frequency is an angle, and it is this angle that is fed into the NCO.
3. The Numerically Controlled Oscillator (NCO)
The NCO generates the sinusoidal signal \( y(t) \). The NCO actually consists of two parts. The first part is an angle counter. This counter generates the angle \( \theta(t) \). It is essentially a counter that counts in the range of 0 to 2\( \pi \).

On each clock cycle, the NCO increments its counter by an amount equal to the value of the loop filter output. That is, \( \theta_j(t) = \theta_j(t-1) + \varepsilon_{\text{loop}}(t) \). Thus the loop filter output represents the change in the NCO output's angle and can be written as \( \varepsilon_{\text{loop}}(t) = \Delta\theta_j(t) \).

Once the PLL has converged, the ideal loop filter output will be \( \varepsilon_{\text{loop}}(t) = \Delta\theta_j(t) = \Delta\theta_x = 2\pi f_x \Delta t \) where \( \Delta t \) represents the amount of time between samples. In other words, once the PLL has converged, the rate of change of the locally generated angle \( \theta_j(t) \) will equal the rate of change of the received signal angle \( \theta_x(t) \).

The second part of the NCO is the sinusoid generator. This part is essentially a lookup table (or series approximation) that outputs the sine or cosine of its input. By connecting the output of the angle counter to the sinusoid generator, the NCO is able to generate the output signal \( y(t) = \cos(\theta_j(t)) \). In practice, the loop filter integrator is often pre-loaded with an estimate of \( \Delta\theta_x(t) \) so that the locally generated signal \( y(t) \) starts out near the frequency of \( x(t) \).

On to Part 2
Now that we've described the basic components of the oscillator in detail, it's time to focus in more detail on phase detector design. In Part 2 of this article, we'll look at ideal and sinusoidal phase detector design. We'll also examine how PLLs can be used to recovery carrier synchronization.

Author's Note: The author would like to thank Kumar Ramaswamy and Paul Knutson (both of Thomson Multimedia) for their help in proofreading this article and also for the numerous discussions on several topics related to this article.

References

About the Author
Louis Litwin is a member of the technical staff with Thomson Corporate Research where he is working on 3G CDMA technology for mobile applications. He received his M.S. degree in Electrical Engineering from Purdue University in 1999, and his B.S. degree in Electrical Engineering (summa cum laude) from Drexel University in 1997. Louis can be reached at