by Dennis Seguine

Simplified FSK Signal Detection

Dennis describes an efficient method for detecting FSK signals using a correlator and analog signal processing in a PSoC. The method is readily applicable to caller ID technology. When combined with a UART, it forms the core of a Bell 202 half-duplex modem.

Frequency shift keying (FSK) is the modulation cornerstone of a number of digital data transmission systems. The transmit signal is fairly easy to generate. Receive processing is tolerant of a wide range of signals and relatively immune to a large class of interfering signals. This article outlines the standard method for detecting FSK signals. I describe a simple almost all hardware solution using the analog signal processing capabilities of a Cypress Semiconductor CY8C27443 PSoC mixed-signal microcontroller.

OPERATING FREQUENCIES

The FSK signal is represented by:

$$v(t) = V_p \sin\{2\pi[f_L + data(f_H - f_L)]t\}$$

f_L and f_H are the respective low- and high-frequency modulation frequencies. The data value is logical 0 or 1. (Alternatively, the low frequency can represent a digital 1 and the high frequency a digital 0, as determined by whichever standard is used.)

Modulation frequencies come in standard pairs for various applications. The example I use in this article has operating frequencies of 1,200 and 2,200 Hz modulated at a 1,200 bps rate. These are typical operating frequencies for Bell 202 modems, caller ID systems, and HART modems.

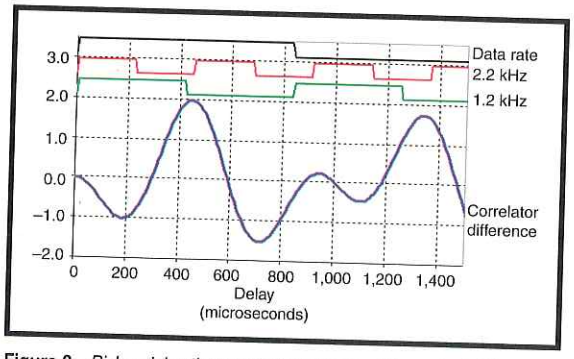

Figure 2—*Pick a delay that maximizes the correlator output difference for the two operating frequencies. Keep the delay less than the bit time at your data rate.*

DETECTION BASICS

There are several basic ways to detect FSK signals. First, let's consider the process of filtering. You can set up band-pass filters at 1.2 and 2.2 kHz, rectify and low-pass filter the results, compare the output levels, and declare a bit "winner." In hardware, this takes a bunch of parts; in software, it takes a fair amount of code and MIPS (we call that "computationally intensive"). Even then, it is level sensitive and not particularly immune to "twist," the telecom guy's name for imbalanced signals, where the low frequency level is typically 1 to 2 dB larger than the higher frequency.

The next standard technique is period measurement. Run the input signal through a zero-crossing detector and then measure the period. This works well when the data rate is slow relative to the lowest FSK frequency; unfortunately, this is not true for a Bell 202. Further, it takes a clock high enough to get decent timing resolution and is not particularly immune to noise.

The example I cover here uses a mix of simple analog and digital hardware. The core element in the FSK

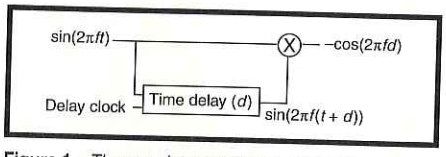

Figure 1—*The correlator multiples a signal by a delayed replica of itself. It can be implemented in hardware (analog or digital) or even in software.*

detector is the correlator, consisting of a delay line and multiplier (see Figure 1). The delay line delivers a time-delayed replica of the source signal. The multiplier multiplies the input signal by this delayed replica.

The mathematics of the multiplier are simple:

$$m(t) = \sin(2\pi ft)\sin[2\pi f(t + d)]$$

Let's start with the simplifying assumption that the levels are unit sized and fixed. Recalling trigonometric identities from high school and applying a little algebra, you find the multiplier output:

$$m(t) = \frac{1}{2}\{\cos[2\pi f(-d)] - \cos[2\pi f(2t + d)]\}$$

The DC value represents the data, and the $2\pi \times 2f$ term (double the input frequency) is eliminated with a low-pass filter. The correlator's multiplier output is a DC level that is a clear function of the operating frequencies and the delay. The product waveforms at the two frequencies are distinctly different. Eliminating the scaling for simplicity, the difference between the high- and low-frequency delay products is:

$$Difference = \cos(2\pi f_L d) - \cos(2\pi f_H d)$$

The purpose is to find a delay that

will result in a multiplier output value for the low frequency that is different from the value for the high frequency (i.e., to find d so as to maximize *Diff*). You can use more of the calculus stuff to take the derivative and find the maximum. A simpler way is to just plot it. Creative use of a spreadsheet or MATLAB shows the correlator output as a function of the delay. An example for $f_L = 1{,}200$ and $f_H = 2{,}200$ is shown in Figure 2.

The *Diff* value should be large, making the signal easier to detect. What constitutes large enough? Because these were defined as unit-sized waveforms, the maximum and minimum signal values after the multiplication operation are 0.5 and –0.5 times the correlator input signal level. You need to set a threshold between these values to determine level detection. You can set the threshold at 0. As a result, any positive value represents one frequency and any negative value represents the other. In this case, the difference must be greater than 0.5. While a difference of 0.25 would be enough, there is nothing to ensure that both values are not positive, say, one at 0.15 and the other at 0.4. In this case, setting up a comparator to find the appropriate nonzero level is less straightforward than a zero-crossing detector.

As a practical matter, a delay that yields the largest possible difference simplifies the filtering and detection process. For the example chosen, the first peak that meets the minimum separation requirement occurs at a delay of 448 µs with a difference value close to the maximum possible of 2.0. Detection with the minimum delay results in the highest detection bandwidth or data rate and allows the most time for low-pass filtering to remove the 2 × carrier frequency term from the multiplier.

CORRELATOR IMPLEMENTATION

The elements of the correlator are easily implemented in the CY8C27x43 PSoC microcontroller. Once out-of-band noise has been rejected, the detection process is a digital problem.

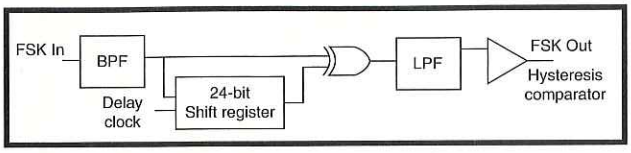

Figure 3—*The implementation of the FSK detector is complete within the PSoC, using mostly standard user module implementations. The shift register and hysteresis comparator are rerouted from standard stuff using software.*

The sine wave output of the filter is converted to a square wave with a comparator. Conveniently, this comparator is built into the band-pass filter user module; it can directly feed the digital blocks.

The time delay function is a simple shift register, with the length and clock set for the delay required, and the sample rate high enough to faithfully represent the waveform.

There are two ways to implement the multiplier. One way is to use an analog modulator. The modulator bit changes the gain of a switched capacitor block from +1 to –1 at the modulation rate (rather like a multiplying DAC going from +1 to –1). The modulator output requires a low-pass filter. This is conveniently done right in the modulator, where the input block of a low-pass filter incorporates the modulator, a handy construct in the switched capacitor block common to almost all versions of the PSoC microcontroller. The signal input of the modulator is the direct signal from the comparator. The modulation signal is selected from the output of the delay line.

Mathematically equivalent, and even easier to implement, an exclusive-OR (XOR) gate acts as a multiplier. The output of the XOR is passed to the same filter that would be used on the modulator version (see Figure 3).

The correlator waveforms and filtered output for both frequencies are shown in Figure 4. The alignment of the waveforms at the same delay value seems pretty clear. For the selected delay (448 µs), the peak difference is positive so that the difference signal is negative at the low frequency and positive at the high frequency. The multiplier output waveform appears as a digital signal with a low duty cycle at f_L and a high duty cycle at f_H. This is filtered to recover the DC level (clearly different for the two modulating frequencies). Now it's just a simple matter routing the blocks, designing the filters, and finding a workable set of clock frequencies to fit both filter and delay line requirements.

PSoC IMPLEMENTATION

My PSoC project is built around a CY8C27x43. I used relatively little of the block, routing, and code resources. With the exception of the comparator, the selected user modules are standard implementations that are well documented and characterized.

The input to the filter is through a programmable gain amplifier (PGA) user module, which allows the addition of gain for a wider input dynamic range. The band-pass is a single-pole pair filter initially designed for 0 dB gain (V/V = 1.0) and centered at the geometric mean of the two modulating frequencies. The bandwidth is set somewhat wider than the frequency difference. The filter is implemented in two switched capacitor blocks and has a balanced loss of about 1.5 dB at both modulating frequencies.

More complex filters for better out-of-band noise rejection can be easily implemented. It's also fairly easy to tap the band-pass filter output and monitor it with a comparator or ADC. You can then use the PGA and the gain adjustment of the band-pass filter to construct an AGC system.

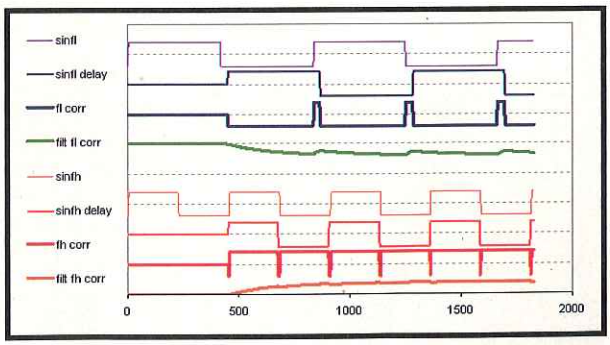

Figure 4—*The fixed delay results in opposite polarity outputs for the two FSK frequencies.*

The band-pass filter's comparator bus output is routed to the input of a digital buffer (DigBuf) user module. The output of the DigBuf is routed to an output row and the adjacent digital block. This enables use of the output row logic look-up table (LUT) to implement the XOR.

There is no shift register user module available in PSoC Designer, so you must creatively connect blocks, the LUT, and routing resources to make one. The shift register is a modification of the pseudorandom sequence generator (PRS) user module.

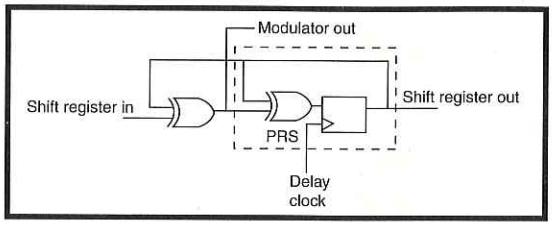

Figure 5—*The shift register is implemented with a pseudorandom sequence (PRS) generator user module and creative use of the global routing and output row logic.*

This user module has selectable feedback taps that are XORed back to the input and normally used to generate a maximum length digital sequence. For this application, the polynomial routes the single tap at the end of the shift register chain back to the PRS input. The PRS is normally hard-wired to invert the output fed back to the input using an XOR. XORing the S/R input with the PRS output yields a shift register by uninverting the input to the PRS's internal S/R. The input connection is modified by adding another exclusive-OR (see Figure 5).

The added XOR is accomplished in the output row LUT. The output of the LUT is routed back to the PRS input using global routing resources to get to the input row. The input row is selected as the data input of the PRS via software, because the PRS data input is normally wired high for the pseudorandom feedback function. The complete routing for analog and digital can be seen in the associated project in PSoC Designer.

The shift register clock is derived from the 24-MHz system clock. With the delay set to 448 µs (selected from data in Figure 2) and a 24-bit shift register, the delay clock is f_{CLK} = 1/448µs/24 = 53.57 kHz divided down by a PWM and the VC1 clock. The length of the shift register is set by selecting the proper tap on the 24-bit PRS using the WritePolynomial API provided as part of the user module. In this case, the maximum length is set with dwPoly = 00800000h.

The modulator's output directly feeds the input of the low-pass filter (see Figure 3). The modulator (XOR) output goes from rail to rail. The signal swing is a function of the degree of match to the required correlator delay and the gain of the low-pass filter. The filter is a two-pole Butterworth at 760 Hz with a rise time of approximately 250 µs.

Added to the correlator delay, the system's response time to a step change in frequency is approximately 620 µs. In order to eliminate false logic triggers caused by noise, a hysteresis comparator is used. The software configures the input routing for

the shift register and comparator, sets power levels, and starts all of the user modules.

DESIGN DEMONSTRATION

I generated a separate project using the CY8C27443 device to provide continuous FSK data for testing the detector in lieu of using an arbitrary waveform generator. The project provides a 1,200-bps signal, two bits high, one bit low, modulating the filtered output at 1.2 and 2.2 kHz. A second test project generates pseudorandom digital data at 1,200 bps, which is then modulated. This is used for more complete testing.

The output of the FSK test generator is biased at analog ground, so the FSK detector input doesn't need additional biasing. Obviously, in the real world, capacitive or transformer coupling to a suitable bias point near analog ground (nominal mid-supply) for the detector would be used.

External components are only required for bypass and input biasing (see Figure 6, p. 22). The input bias connection will be slightly different if the output of a transformer-coupled DAA is used.

The digital source (modulating waveform), FSK output, intermediate waveforms, and digital output are shown in Photo 1. Trace 1 shows the test source digital modulation at a 1,200-bps signal, two bits high, one bit low. Trace 2 shows modulated 1,200- and 2,200-Hz FSK signals. This is easy to see because of the wide spread in frequency. Trace 3 shows the output of the band-pass filter's comparator. Trace 4 shows the correlator delay line output. The phase shift between the comparator and delay line outputs at the two operating frequencies is clear to see.

Trace 5 shows the XOR output. The duty cycle of the waveform clearly lines up with the phase relationship of the signals. Trace 6 shows the filtered correlator signal. This recovers the DC average output of the XOR. More of the ripple can be taken out at the cost of additional processing delay.

Trace 7 shows the detection comparator output. With just a little bit of hysteresis, the digital output is stable and noise-free over a wide range of input levels. The addition of gain in the PGA could push the input sensitivity to smaller signal levels for remote applications, such as a HART modem.

The detection delay time is approximately equal to the RMS sum of several delays, including input band-pass

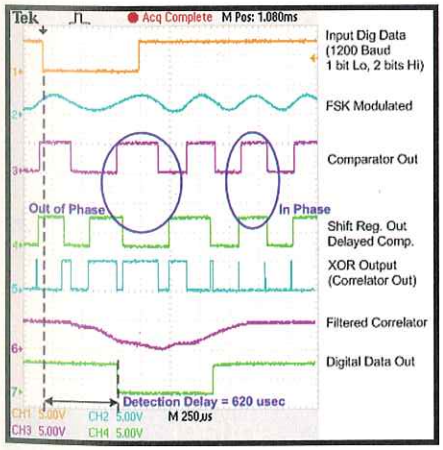

Photo 1—*The waveform stack-up shows the time delay in the shift register, correlation, filtering, and final recovery of digital data. No, this is not the new Tektronix seven-channel oscilloscope; it's a composite of two images.*

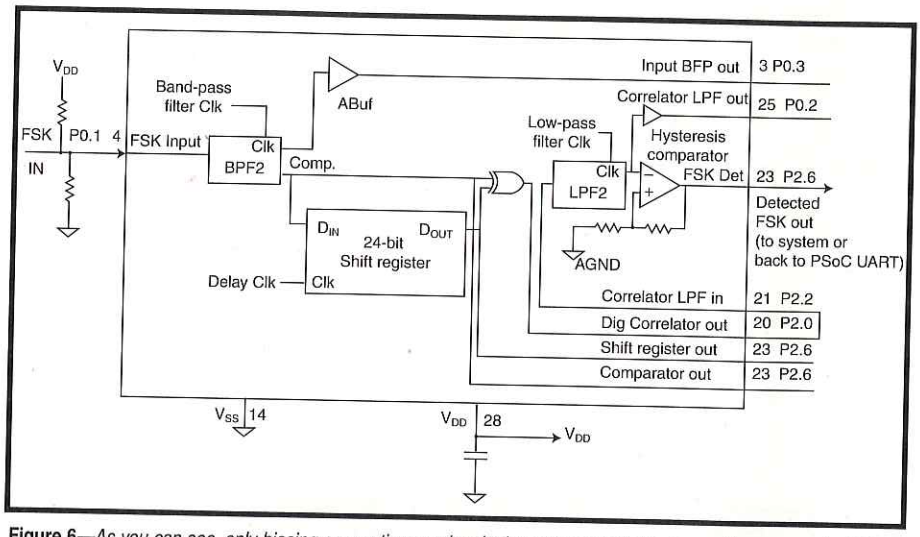

Figure 6—*As you can see, only biasing connections and a single point-to-point chip connection are required. The output can feed your UART, or you can use the one still available in the PSoC.*

filter response, correlator delay, and correlator low-pass filter response. The total detection time is about 620 µs, a good deal less than the bit time of 833 µs for the 1,200-bps signal.

The standard data presentation for demodulation is the eye diagram shown in Photo 2. Pseudorandom data (Trace 1) is fed to the test modulator and then into the FSK detector. The output of the demodulator is measured for some extended period. The quality of the demodulation is measured by the "openness" of the "eye." This technique clearly shows robust performance. It can be made a little better, if absolutely necessary, by increasing the sample rate on the band-pass filter, adapting the delay line to the higher sample rate and increasing the sharpness of the low-pass filter. This would increase the required block count and power consumption.

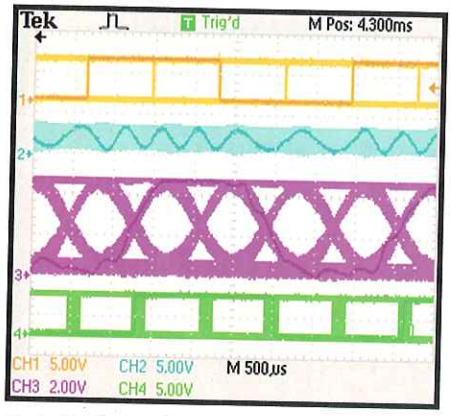

Photo 2—*The eye diagram constructed from multiple passes of pseudorandom data shows clear signal recovery.*

DESIGN OPPORTUNITIES

I've described a bare essential FSK detector design. There are numerous opportunities for design improvements, including increasing the gain, improving the filter selectivity, adaptating to other operating frequencies, and completing the connection of the recovered serial digital data via a UART RX to the CPU's bus. The design techniques are easily extended to these other applications.

DESIGN ALTERNATIVES

You're a digital or software designer, and you don't like my analog. You want an algorithm that you can stuff into that new ARM development kit or 100-MIPS DSP chip you paid all that money for. So, here's the algorithm (a description, not the code).

First, digitize (a terrible thing to do to a perfectly good analog waveform). Don't believe Nyquist, set up the sample rate for the filter at a frequency equal to an integer multiple of both FSK signals. In this case, 13.2 ksps works (11× on 1,200 Hz and 6× on 2,200 Hz). Higher frequency is better still. Use enough bits to make the filter work over your expected input signal range.

Next, implement band-pass filter 1-dB flatness from 1,200 to 2,200 Hz. Build a comparator on the filter output. The zero-crossing comparator is simply the polarity of the band-pass filter output.

Build a delay line with a FIFO. The length is 448 µs × sample frequency. (It's six in this case.) Adjusting the filter and delay line sample rates so that they are synchronous can provide a more accurate delay and cleaner correlator "waveforms." Following this, multiply comparator and delay signal single bit values using logical single bit XOR. Low-pass filter the product (the XOR output) using an infinite impulse response (IIR) filter.

Finally, set thresholds and make a hysteresis comparator. You now have serial digital data. Connect it to your software UART. The implementation of caller ID is a simple matter of following earlier articles. This is essentially the technique Rick Hood implemented for his 300-bps, PSoC-based modem ("PSoC 5-Cent Modem," *Circuit Cellar* 146, September 2002).

EASY IMPLEMENTATION

This project demonstrates bare essential FSK signal detection. It is immediately adaptable to caller ID and Bell 202 modem (1,200 bps, half duplex) applications. It uses less than half of the analog capability of an CY8C27*x*43 microcontroller, most (but not all) of the digital capacity of the chip, only 14 lines of source code in C, no run-time manipulations, and no interrupts. I tested this technique successfully at 1,200 bps in PSoC power line modems at carrier frequencies of 132.45 kHz.

FSK detection using the correlator is the standard way to go. It's easy to implement in hardware (a Cypress PSoC and no external parts) or in code.

Dennis Seguine (seg@cypress.com) is an itinerant bluegrass musician and goat farmer who has funded his real life by working for Cypress Semiconductor since August 2000. He has five patents on critical-care medical devices and numerous patents pending on the Cypress PSoC chip, software, and applications.

SOURCE

CY8C27443 Microcontroller
Cypress Semiconductor Corp.
www.cypress.com