CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

FEATURE ARTICLE

by Neal Martini

Signal Generation Solution

Build an Inexpensive RF Signal Generator

Tired of going to a local university lab to use a signal generator, Neal designed his own. In this article he explains how he built the PIC16F877A-based controller and RF module.

I worked at home for a year and a half on a project that required a frequency source that generated sine wave signals in the 100- to 375-MHz frequency range. I had access at a local university to high-quality Hewlett-Packard signal generators, but going back and forth to the lab was a bit inconvenient. My first thought was to look on the Internet for an affordable signal generator to suit my needs. Needless to say, only expensive generators were available.

With many years of professional and hobby experience in lower frequency digital and analog systems, I decided to venture out and build my own RF signal generator. I realized that things were a little trickier at higher frequencies, but I was confident it would be fairly straightforward. I had some experience with low-frequency narrowband mixing circuits, so I thought I had a pretty good handle on frequency translation using mixers. So, I proceeded with reckless abandon. Boy, did I ever underestimate the task I was about to undertake!

The more I dug in, the more I realized that almost everything is different when working at RF. The terminology (e.g., VSWR, compression points, frequency pulling/pushing, intermodulation distortion, and dBm) was foreign to me. Much more attention was given to design details like complex impedance matching and signal path leakage. The way you breadboard, lay out the PCB, and test were—you guessed it—all different. Even the parts suppliers were companies that I wasn't familiar with. And finally, and most importantly, the difference between theory and reality was dramatic.

Nevertheless, it was a great learning experience. If you want to expand your electronic horizons a little, this is a good project to try. Before I describe my design, I'll cover some of the things I learned that might help you understand some of the trade-offs I made.

IMPEDANCE MATCHING

Voltage standing wave ratio (VSWR) is an important term to understand when it comes to impedance matching at RF. In it's simplest form, a VSWR = 1, which is sometimes written as 1:1, means that the source and load have a matched impedance such as a 50-Ω source driving a 50-Ω load. If a purely resistive load impedance were twice that of a purely resistive driving source impedance, as would be the case if a 50-Ω resistive source were driving a 100-Ω resistive load, then the VSWR would be 2:1.

So why so much focus on impedance matching at RF? At RF, if there's an impedance mismatch, some of the signal gets reflected back from the load to the driving source. A consequence is that some of the power that you wanted to deliver to the next stage in the circuit is reduced. For example, if you have a VSWR of 5:1, you lose 2.55 dB of your transmitted signal.

Another consequence is that the signal reflected back to the driving source can cause problems. For example, with a VSWR of 5:1, 44% of the power is reflected back to the source. In a high-power amplifier, this reflected signal could be so strong that it damages the amplifier's output stage.

Another example is when you have a mismatch at the output of a mixer. The reflected signal gets sent back into the mixer and is remixed. This could produce spurious outputs. A table showing some typical VSWR values and what the power loss/reflection would be for various mismatches is available, along with some other useful items, on the *Circuit Cellar* FTP site.

As it turns out, most RF parts don't have the typically used characteristic impedance of 50 Ω. In fact, some parts are dramatically different. Take a simple low-pass filter for example. In the passband for the filter, the VSWR is typically 1.1. Pretty good! Well, not so fast. In the stop-band the VSWR of the filter typically can be 20:1 or worse. What that means is that almost all of the power (82%) outside the passband of the filter is reflected back to the driving source. If the filter were at the output of a mixer, the consequences of this mismatch could cause significant problems in the mixing process.

So, what can you do if most parts aren't the nominal 50 Ω and you can't live with the consequences? There's plenty of literature available about using L/C networks to match impedances for narrowband designs. However, in wide-band systems like the signal generator described in this article, a common technique is to use resistor pads between the mismatched impedances. Because they are purely resistive, they work their magic over a broad frequency spectrum.

What happens with a resistor pad is that the signals that eventually will be reflected back from the load will be attenuated on the path forward to the load and on the return trip. This effectively alters the VSWR as seen by the source because a lower reflected signal appears back at the source. Of course, the VSWR is improved at the expense of attenuation to the wanted signal as well. This is tolerable in many cases, as you'll see in the RF generator design I'll describe later.

The VSWR improvement using resistor pads can be dramatic. For example, a VSWR of 20:1 can be changed to a VSWR of 1.59:1 with the addition of a 6 dB pad. A table showing how various VSWR values can be changed with different pads is posted on the FTP site.

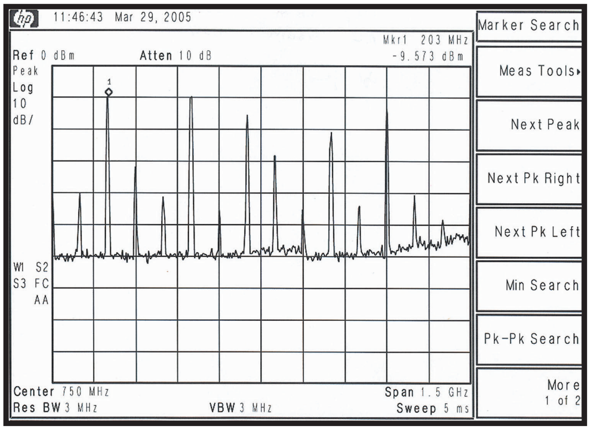

Figure 1—*Take a look at an example of the frequency spectrum at the IF port of a real world DBM mixer. The LO signal and the RF signal were provided by two commercially available signal generators (F_{LO} = 500 MHz and F_{RF} = 700 MHz). The output spectrum is loaded with spurs that would cause problems in a wideband system.*

REAL MIXERS ARE MESSY

A classical mixer has two input ports (RF and LO) and one output port (IF). Ideally, the sine wave F_{LO} at the LO port modulates the signal F_{RF} at the RF port, and the output port IF contains $F_{IF} = F_{LO} \pm F_{RF}$. In the commonly available double-balanced mixer (DBM) used in this project, this result is true; however, the output also contains undesired spurious outputs (called spurs) that can cause distortion if they aren't handled carefully.

Why are so many spurs produced? For one thing, the mixing action in a DBM is achieved by turning various diodes on and off to produce modulation. This process is called biphase modulation. The mixer's output can be mathematically modeled as follows:

$$V_{OUT} = V_{RF}\sin(w_{RF}t)\left(\frac{4}{\pi}\right)\sum_{ODD}\left(\frac{1}{n}\right)\sin(nw_{LO}t) \qquad [1]$$

With a DBM you're basically modulating the RF signal with a series of sine waves that are odd harmonics of the LO, each producing signals at the mixer's output.

Let's look at another source of unwanted mixer output components. Real world RF mixers have signal leakage between all the three ports. This causes signals to appear at the output IF port located at the RF and LO frequencies. According to classical communication theory, this isn't supposed to happen in DBMs. This leakage is unavoidable for RF signals.

The final source of spurs is generated because the conducting diodes are nonlinear. Although the nonlinearity is required to produce the wanted sum and difference frequencies, the higher order nonlinearities produce unwanted spurs.

The net result is that the mixer produces output at various frequencies:

$$F_{IF} = \pm mF_{LO} \pm nF_{RF} \qquad [2]$$

where *m* and *n* are integers. Figure 1 shows how catastrophic this can be. It shows the output of a spectrum analyzer that was attached to the IF port on a DBM mixer. The LO and RF signals were provided by two commercially available signal generators. F_{LO} equals 500 MHz. F_{RF} is 700 MHz. As you can see, the output spectrum is loaded with spurs that would cause problems in a wideband system.

Hittite's spur calculator is an excellent tool for predicting a DBM's output (www.hittite.com). If you specify the frequencies and the levels of the RF and LO ports, the tool will show you the frequencies at which all of the spurs will occur and what their power levels will be.

Figure 2 shows an example of the output from the Hittite tool. In this example, the LO is 1.5 GHz and the RF signal varies from 1.5 to 2 GHz. If, for example, you want to know which mixer IF outputs will occur if the RF signal is 1.7 GHz, you go to 1.7 GHz on the RF frequency axis and move up the graph vertically. Every line you cross is another output that will be at the IF port. The level of each output signal is given in the boxes below the graph.

You can use this tool to come up with a frequency plan for your design. It will help you choose things like the frequencies and drive levels required for the LO and RF oscillators in order to get the IF results you want. It will also help you plan which filtering is required around the mixer.

PROTOTYPING & TESTING

When there are 2-GHz signals running around in a circuit, you have no choice but to prototype with a PCB. I tried hand soldering and wire wrapping, but the results were chaotic.

You can get away with a two-layer PCB design as I did for this project, but four layers are better. You have to use surface-mount devices to achieve quality results.

At RF frequencies, components with axial leads add too much inductance and capacitance to be practical. I refused to go any smaller than 0805-size SMT devices because that was the smallest I could handle with tweezers and magnifying goggles. It seemed to work fine.

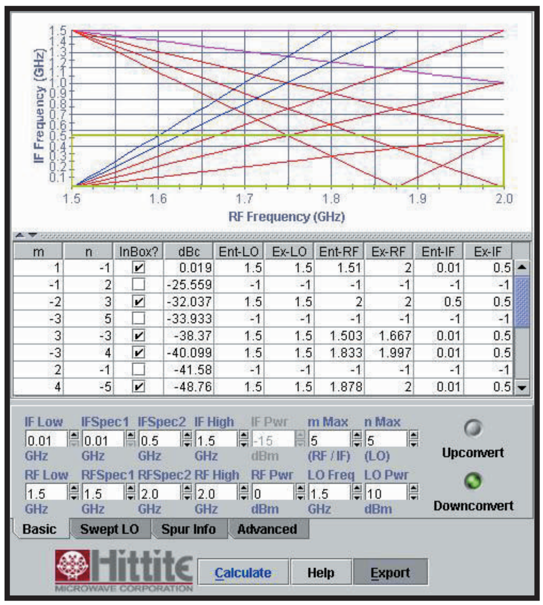

Figure 2—*The LO signalis 1.5 GHz. The RF signal varies from 1.5 to 2 GHz. The graph shows the different IF port outputs that will occur as the RF frequency is varied. The level of each output signal is given in the boxes below the graph.*

I resorted to water-soluble solder paste and a toaster oven for the reflow. I was amazed at how well this simple approach worked. I controlled the toaster oven's temperature and timing manually.

Laying out a PCB at these frequencies can be tricky. Given the trace thickness used by your PCB supplier, you need to use trace widths that provide a 50-Ω system impedance. I needed approximately 100-mil trace widths for my PCB.

In addition to controlling trace widths, try to keep the main signal path as close to a straight line as possible to minimize any spurious modulations. If you need to make a right angle, it's recommended that you do it with two 45° steps. Even the size and length of a via is a factor at these RF frequencies, but I chose to ignore this precaution without apparent consequences.

It's also necessary to put an RF circuit in an RF tight enclosure in order to keep external signal sources out and the generated signals in. RF tight means no open holes in the box to pass wires through (they would leak RF like crazy). It also means shielded connectors in and out for signal lines and pass-through capacitors for filtering DC supply lines. I obtained pass-through capacitors from a surplus house for $1.50 each. It's worth searching for these because they can be pretty costly.

Incidentally, if you study the information available online, you'll find that standard BNC connectors work well at 2 GHz. The BNC connectors and cables are cheaper and more readily available than the higher frequency SMA counterparts.

A final note on testing and debugging an RF circuit. It's best if you hard-wire shielded cable to test points rather than use pigtail leads. The parasitics and extraneous coupling that exist with unshielded lengths of wire will produce confusing results.

Enough about all the gotchas! It's time to move on to the RF signal generator's design.

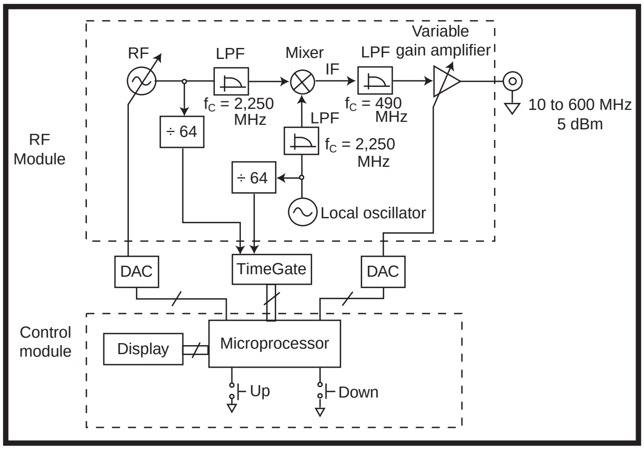

Figure 3—*A voltage from the microprocessor controls the RF signal frequency. The actual RF and LO frequencies are measured by the microprocessor. The microprocessor calculates the actual IF frequency and displays it. A voltage from the microprocessor controls the amplifier gain to maintain the desired output level.*

ARCHITECTURE

My goal for this project was to design a signal generator that produces sine waves from 10 to 600 MHz at a constant output power level of 5 dBm. Let's take a look at how I did it.

The assumed load is 50 Ω, which is typical for RF systems. Talking about signal levels in terms of decibels relative to 1 mW (dBm) is common when you're dealing with RF systems. Equation 3 is for converting from power in watts to power in dBm.

$$\text{Power}_{\text{dBm}} = 10\log\left(\frac{\text{Power}_{\text{watts}}}{1\text{ mW}}\right) \qquad [3]$$

A table showing the relationship between power in dBm, RMS voltage, and power in milliwatts is posted on the FTP site. The table assumes a 50-Ω system.

The 5 dBm design specification for this generator is equivalent to a 0.4-V RMS sine wave. This will produce 3.2 mW when driving a 50-Ω load.

Figure 3 shows the overall architecture for the signal generator. It consists of two main modules. The RF module produces the 10- to 600-MHz 5-dBm signal. The controller module directs the RF module's actions.

The signal flow in the RF module is a straightforward mixing process. The local oscillator is fixed at approximately 1.5 GHz. The RF oscillator varies from 1.5 to 2.1 GHz depending on the output frequency you want from the generator. A voltage from the microprocessor sent via a D/A converter controls the RF oscillator's frequency.

I chose the high operating frequencies for the LO and RF oscillators in order to keep unwanted spurs from appearing in the desired IF output range of 10 to 600 MHz. Next, the outputs of the RF and LO oscillators were frequency divided down so the microprocessor could measure them. These divided-down signals were 23- to 32-MHz digital signals; they were routed to the microprocessor through time windowing control logic.

Before the RF and LO oscillator outputs are mixed, the signals are low-pass filtered to attenuate any harmonics present at the oscillator outputs. The RF and LO signals are then mixed by the mixer and an assortment of signals appear at the IF output. This IF output is then low-pass filtered to eliminate most of the unwanted spurs. Following this, a variable-gain amplifier whose gain is

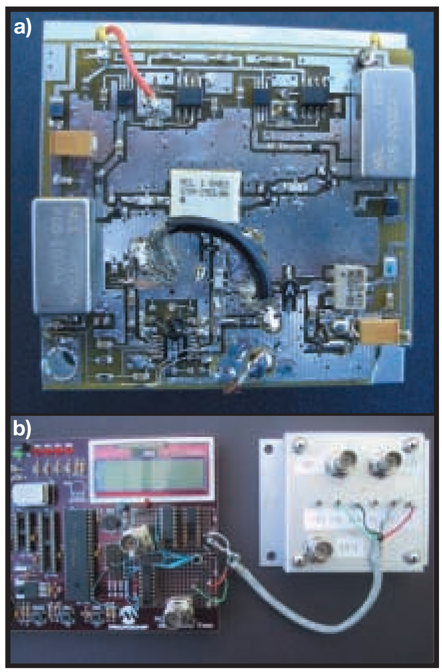

Photo 1a—*The PCB is approximately 2.5″ on a side. It contains mostly SMD devices. The DBM mixer is in the center, and the two VCOs are located on the left and right sides. The top part of the board is the divide-by-64 circuitry. The lower part contains the IF amplifiers.* **b—***The modified demo board on the left is attached to the RF module in its RF tight enclosure. For the sake of clarity, I omitted the shielded cables normally located between the RF and LO BNC connectors on the RF module and the controller.*

controlled by the microprocessor amplifies the RF signal. This analog gain control signal is supplied via a D/A converter driven by the microprocessor. An LCD shows the generator's output frequency. The up and down push buttons enable you to select the desired generator frequency.

RF MODULE

The RF module PCB is shown in Photo 1a (p. 15). Figure 4 is a detailed schematic of the module.

To begin the design, I used the Hittite tool to determine where in frequency I wanted to operate to minimize spurs. I chose to operate the mixing process between 1.5 and 2 GHz.

The first component I selected was a Mini-Circuits SYM-25DLHW mixer mainly because of its operating frequency range. I then chose a POS-2000A voltage controlled oscillator (VCO) for the RF and LO oscillators. The POS-2000A's output frequency range is approximately 1.3 to 2.1 GHz.

Applying 0 to 20 V to the V_{TUNE} input controls the frequency. I selected this VCO not only for its operating frequency range, but also for its output level. It turns out that if you want good results from a mixer, you need to drive it at the signal levels it was designed for. The SYM-25DLHW mixer is designed to have a 10-dBm LO signal level and an RF signal level that's at least 10 dB below the LO (or around 0 dBm). The POS-2000A has an output level of 10 to 12.6 dBm, which makes it an excellent choice for both the LO and RF oscillators.

The LO VCO's output passes through a small 1-dB pad to help better match impedance to the low-pass filter. I would've liked to have used a larger pad for better matching, but I needed to keep the LO signal level to the mixer at the 10 dBm design specification.

The LO signal then feeds into a Mini-Circuits LFCN-2250 low-pass filter. The filter's 3-dB cutoff frequency is 2.525 GHz. These LFCN-series filters are really slick seven-section filters contained in a tiny ceramic package. They take up little board space and minimize the headaches caused by the parasitics associated with filters constructed with separate SMT components. The output of the LFCN low-pass filter drives the mixer's LO port.

The output of the LO VCO is also tapped off via a 475-Ω resistor in order to feed a UPB1507 prescaler. The input to the prescaler must be between –15 and 5 dBm to operate properly. The UPB1507 prescaler divides the analog signal by 64 and outputs a 1.6-V_{PP} sine wave like output. This output is transformed into a clean digital signal by the high-speed TL714 comparator.

The RF signal path is similar. The main difference, as I mentioned earlier, is that the SYM-25DLHW mixer is made to operate with an RF input level at least 10 dB below the LO level. Two 6-dB pads are included to provide this attenuation. As an additional benefit, they provide excellent impedance matching in the RF signal path.

Incidentally, the pads in this circuit are also made by Mini-Circuits. They're extremely compact and take up little board space. As you can see, I used a lot of Mini-Circuits components. The company is a quality supplier. The literature it provided was excellent, and the specifications available on the company's web site were thorough and accurate.

There is a pad at the mixer output for impedance matching. Matching at the mixer output is critical. The signal then passes through another LFCN-series low-pass filter with a 3-dB cutoff of 650 MHz. The filter's output feeds into an Analog Devices AD8367 variable gain amplifier, the purpose of which is to maintain the required 5-dBm output signal level as the frequency varies. A control voltage (0 to 1 V) on the AD8367's gain pin

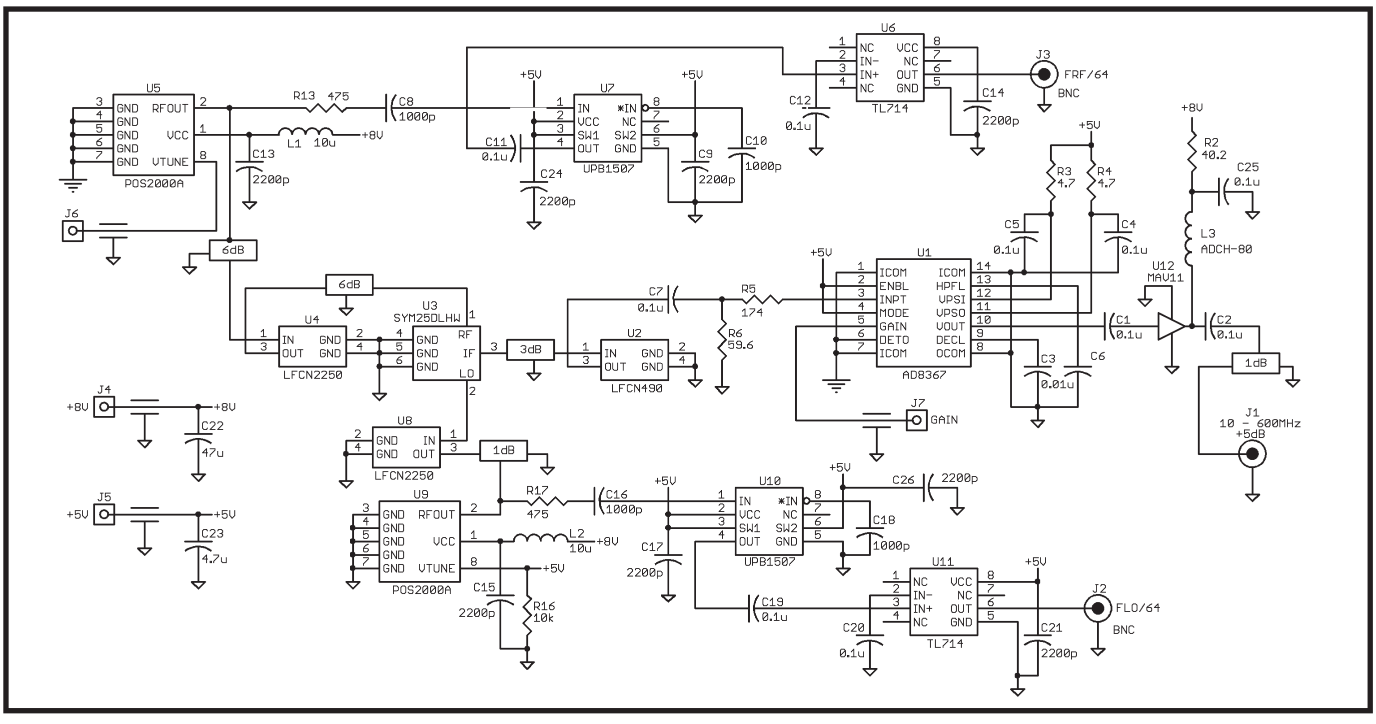

Figure 4—*The two POS-2000As are used to generate the RF and LO signals in the SYM-25 DLHW DMB mixer. The variable gain is accomplished by the AD8367. The UPB1507 prescalers provide the frequency division function.*

varies the gain from –2.5 to 42.5 dB. The AD8367's output is then amplified once more by a stage that uses a Mini-Circuits MAV-11 MMIC integrated amplifier. This provides a robust interface to the outside world. It's included because it's cheaper and easier to replace than the AD8367 if someone like me abuses the signal generator's output.

Table 1 shows the expected signal levels at various points throughout the RF generator. Two operating frequencies are shown. These values were derived from the detailed specification sheets of the various components. The gain as a function of frequency was initially set using these values, but the measured output was consistently low by 3.2 dBm. I added a 3.2-dBm constant to the gain function, which enabled the microprocessor to keep the RF signal generator's output at 5 dBm. The gain function is a straight line fit to this adjusted data.

	Signal levels (dBm)	
	RF = 1.6 GHz	RF = 2 GHz
VCO Output	11.5	10.14
LFCN-2250 Output	5.17	3.63
Mixer RF input	–0.83	–2.37
Mixer IF output	–0.751	–9.82
LFCN-490 Output	–10.73	–13.68
AD8367 Input	–22.13	–25.08
Mav-11 Input	–7.6	–7.16
RF Gen output	5	5

Table 1—*The expected signal levels at various points throughout the RF generator were derived from the specification sheets for the various components. Two different RF operating frequencies are shown.*

CONTROLLER MODULE

The controller module is shown in Figure 5. I implemented this controller on a Microchip PICDEM 2 Plus demonstration board. The schematic shows only the portion of the demonstration board that I used. I used the prototype area to add the circuitry for controlling the RF module. The additional 8- and 20-V power supply designs are included in the schematic for completeness, but bench supplies were used for the prototype. Photo 1b shows the modified demonstration board attached to the RF module in its RF tight enclosure.

The controller includes a Microchip PIC16F877A microcontroller running at 4 MHz. The MCU's job is light in this application, so its limited amount of horsepower didn't matter too much.

As you can see in Figure 5, I included a two-channel MCP4922 DAC to supply the control voltages to the RF POS-2000A VCO and for the gain control to the AD8367. The DAC's outputs were buffered using the TLE2142 low-noise op-amp. I initially attempted to use the PIC16F877A's built-in PWMs with low-pass filtering to produce the control voltages, but I wasn't happy with the output's purity. It turns out that the VCO and variable gain amplifier are extremely responsive to fluctuations on these control lines. The buffered DAC with the associated MCP1541 reference source perform quite well. They provide clean, stable signals to the control lines.

The LO and RF frequency measurements are made in a unique way. The challenge is to measure a signal that's at approximately 30 MHz with a microprocessor running at 4 MHz.

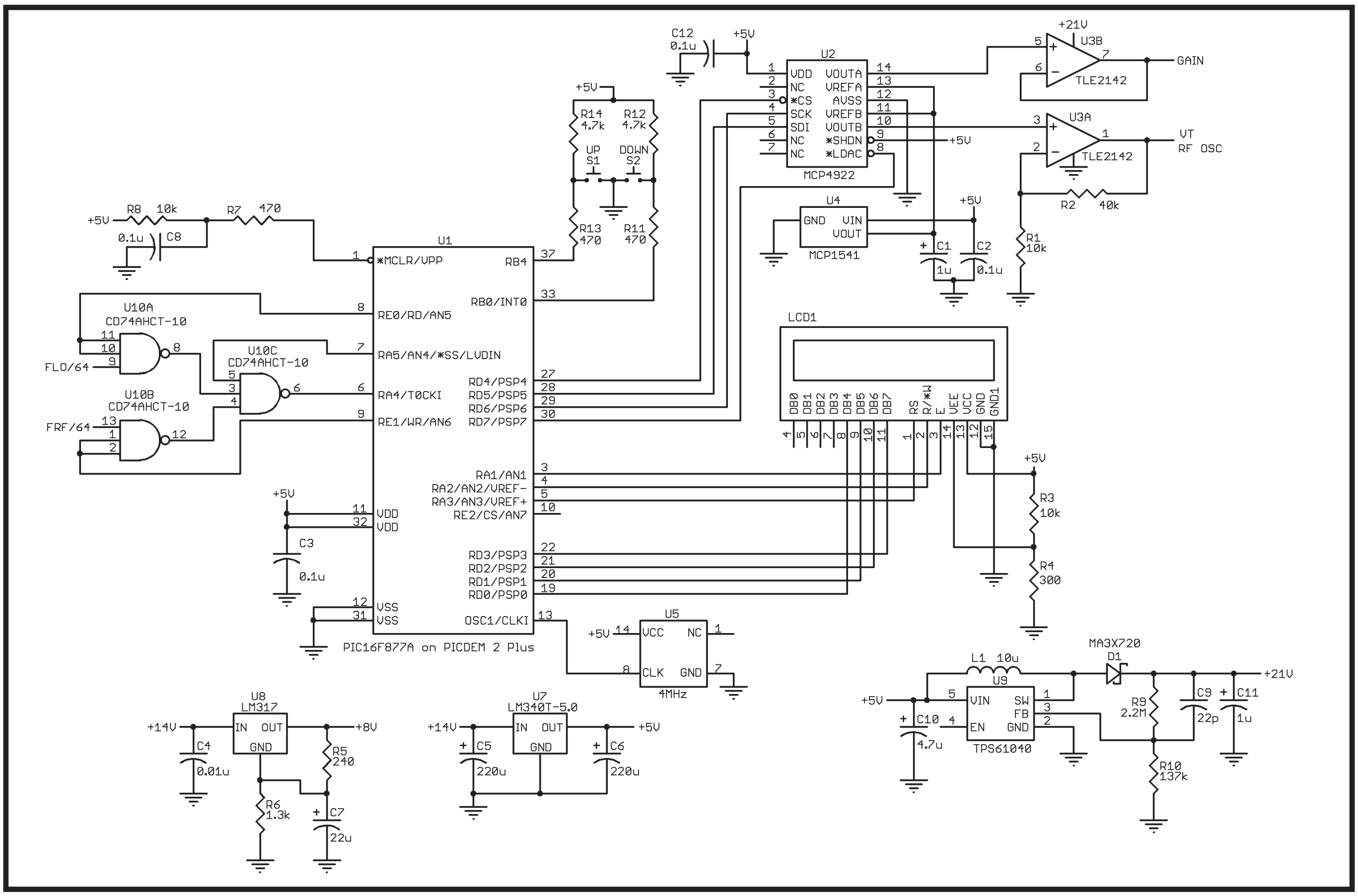

Figure 5—*The controller provides the analog voltages for RF frequency control and IF signal amplifier gain. The measurement of the actual RF and LO frequencies is also performed in the controller.*

This design uses the PIC16F877A's internal Timer0 with a built-in prescaler. The prescaler is specified at 10-ns rise and fall times, which enable it to be clocked by an external source at up to 50 MHz. I gated the input to Timer0 for 1 ms and then used Timer0's contents to calculate the frequency.

The process seems pretty straightforward, but there was a problem. Timer0 with its prescaler is 16 bits long, but the microprocessor can read only the top 8 bits. To determine the prescaler's contents, I used the microprocessor to pulse Timer0 until the prescaler overflowed into the upper 8 bits of Timer0. The number of times you have to pulse to get to overflow enables you to identify the prescaler's initial content. The PBasic pseudo code used to execute this is shown in Listing 1 (p. 20). A simple flowchart is posted on the FTP site.

A standard LCD and two push buttons are included. I tried to keep the interface simple, so the only controls are an Up/Down frequency button. In response to a request to change the frequency, the microprocessor will increase or decrease the control voltage to the RF VCO. The output frequencies of the RF and LO VCOs are then measured in the aforementioned fashion, and the IF output frequency is calculated. Following this, the microprocessor adjusts variable gain amplifier's gain as a function of frequency. The microprocessor then displays the IF output frequency on the LCD.

GOOD OUTPUT?

A table showing an output comparison between the RF signal generator and a commercially available generator is posted on the FTP site. Both generators were set to deliver a 200-MHz, 5-dBm signal.

All unwanted spurs of any consequence are pushed out of the 10- to 600-MHz range of the generator except for the pesky second harmonic of the desired output signal. As I predicted with the Hittite tool, this component is down approximately 36 dB relative to the wanted fundamental.

The expensive spread is better, but I didn't do too badly with respect to harmonic levels and total harmonic distortion. Furthermore, the RF generator's output levels were well maintained across the frequency range to 5 dBm (±0.5 dBm). A sample plot of the output spectrum is available on the FTP site.

NEXT STEPS

This was a fun and challenging project, but I feel like I've just seen the proverbial tip of the iceberg. I now want to evolve this design and continue exploring the process of working at these higher frequencies. I plan to control the LO and RF oscillators' frequencies with phase-locked loops for better stability. I would like to use VCOs that operate at 5 V and have control voltages of up to 5 V to simplify the necessary power supplies. I'm going to look for a wider BW variable-gain amplifier in order to operate at higher frequencies. I would also like to add true RMS detection to the gain control loop for output level accuracy. Finally, I'll probably add a full keypad interface to the microprocessor in an effort to create more flexibility when controlling the frequency and level. ■

Neal Martini holds an M.S.E.E. from the University of Missouri, Rolla. He is now retired after working 24 years for Hewlett-Packard in the Inkjet and LaserJet printing businesses. Neal is currently pursuing several independent product developments in a variety of application areas. In addition to playing racquetball and golf, he enjoys woodworking and traveling with his wife. You may contact Neal at nealmartini@cableone.net.

PROJECT FILES

To download the code and additional files, go to ftp://ftp.circuitcellar.com./pub/Circuit_Cellar/2005/182.

RESOURCES

K. Maxon, "Have You Seen My New Soldering Iron?" Encoder, Seattle Robotics Society, www.seattlerobotics.org/encoder/200006/oven_art.htm.

PCB Impedance Calculator, www.emclab.umr.edu/pcbtlc/microstrip.html.

SOURCES

AD8367 Variable gain amplifier
Analog Devices
www.analog.com

Spur calculator
Hittite Microwave Corp.
www.hittite.com

MCP4922 DAC and PIC16F877A MCU
Microchip Technology
www.microchip.com

LFCN-2250 low-pass filter, MAV-11 MMIC integrated amplifier, POS-2000A voltage VCO, and SYM-25DLHW mixer
Mini-Circuits
www.minicircuits.com

Listing 1—*The PBasic pseudo code shows how an approximate 30-MHz signal is measured using a relatively low-power 4-MHz microprocessor. The design uses the PIC16F877A's internal Timer0 with a built-in prescaler.*

```
philo   var word              //Contains RF frequency after measurement
measrf:
        High PORTE.1          //Enable RF oscillator gate
        Low PORTA.5           //Disable t0ckl gate
        LowPORTE.0            //Force local oscillator gate output high
        TMR0=0                //Clear TIMER0
        PULSOUT PORTA.5,100   //Enable tockl gate for 1 ms
        Low PORTE.1           //Disable RF oscillator gate
        fhilo.byte1=TMR0      //Extract upper 8 bits of Timer0
        call extract          //Extract value in prescaler
        RETURN
extract:
            For I=1 to 255    //Pulse Timer0 until prescaler overflows
            PULSOUT PORTA.5,1
            If TMR0-fhilo.byte1=1 then goto done //Is there overflow?
            Next I            //No
done:   fhilo.byte0=256-I     //Yes. 256 - number of loop cycles =
                              //prescaler value
        Return
```