

***Authors' draft of article published in
March/April 2004 QEX/Communications Quarterly***

Please refer to <http://www.qsl.net/ke5fx/synth.html> for the latest updates and errata

A Versatile Hybrid Synthesizer for UHF/Microwave Projects

John Miles KE5FX

Richard Hosking VK6BRO

Ever want to build a “DC to daylight” receiver or digitally-controlled spectrum analyzer? How about a transceiver for the Amateur UHF bands, or a signal generator with octave-band coverage? At the heart of each of these projects is a local oscillator with good stability and spectral purity. Our goal in this article is to present a versatile and *practical* synthesizer design that can address almost any homebrewer’s need for a digitally-tunable signal source from VHF to 4 GHz.

Unlike most published approaches, we’ve focused our design efforts as much on affordability, flexibility and reproducibility as on “specsmanship.” The synthesizer we’ll describe offers continuous coverage between 1 and 2 GHz with fast switching time, sub-1 Hz tuning resolution and very good close-in noise performance. There’s nothing to tweak or align with exotic test equipment – if you build it, it will work. Best of all, every component is available off-the-shelf from Mini-Circuits, Digi-Key, and Analog Devices for a total of less than US \$200.00!

Design Overview

When it comes to modern synthesizer design trends, numerous authors have made the case for a hybrid topology that takes advantage of the combined strengths of direct digital synthesis (DDS) and traditional phase-locked loop technology.^{[1](#)[2](#)[3](#)[4](#)[5](#)} Before these hybrid techniques became practical, PLL synthesis often involved awkward tradeoffs between frequency step size, freedom from spurious outputs, and overall loop complexity. While standalone DDS chips do not yet have the output frequency range (and in many cases, the spectral purity) to offer an across-the-board replacement for PLL technology, it's easy to use DDS technology to build a PLL synthesizer with arbitrary frequency-control precision, competitive spectral purity, and low overall complexity.

In its most basic form, a hybrid synthesizer uses a DDS source to provide a stable, clean, and precisely-tunable reference for a conventional PLL. The output of a hybrid synthesizer derives its tuning precision and stability from its DDS reference while providing the frequency coverage range typical of a PLL. The approach we've taken is similar to that in [Cornell Drentea, KW7CD's recent microwave DDS-driven PLL article](#)⁶, adapted to use newer, more readily-available parts and improved post-DDS filtering. Tuning control is provided via the PC parallel port or an Atmel microcontroller. Like Cornell's design, our synthesizer can be used as a standalone VHF/UHF/microwave source or with an external frequency divider to achieve exceptional noise performance in HF applications.

The synthesizer's heart is the AD9852 DDS chip from Analog Devices. The AD9852 is used to generate a finely-tunable reference signal near 10.7 MHz for the PLL. Chosen for its superior performance over the less-costly (and somewhat more popular) 10-bit AD9850, the 12-bit AD9852 is clocked by either an onboard 10 MHz oscillator or an external 10 MHz source, using the chip's clock-multiplier feature to generate internal clock frequencies between 80 and 120 MHz.

After passing through a crystal filter to tame any wideband spurs, the signal from the DDS is amplified and converted to a square wave by an [LT1016](#) comparator from Linear

Technologies. The filtered and squared PLL reference signal from the comparator, along with a sample of the signal from the synthesizer's VCO obtained from a resistive divider, is applied to any of several programmable PLL synthesizer ICs from National Semiconductor's [LMX2306/16/26](#) or Analog Devices' [ADF4110/11/12/13](#) family. These PLL chips are marketed toward the wireless- and cellular-communications industry, but their low prices, ease of use, and availability in small quantities make them attractive for Amateur Radio applications from the lower VHF region to frequencies well beyond the 2.4 GHz band.

Generation of the synthesizer's output signal is handled by a commercial varactor-tuned voltage-controlled oscillator module. A wide variety of prepackaged VCOs are available with industry-standard PC board footprints from vendors such as Mini-Circuits and Synergy Microwave, offering various tuning ranges, output power levels, harmonic and noise performance specifications, and supply voltages. The synthesizer project described here uses the [ROS-2150VW](#) from Mini-Circuits, which provides an impressive 970-2150 MHz tuning range for about \$30.

Overall loop characteristics such as phase margin, bandwidth, and lockup time are determined by a third-order active loop filter based on the low-noise [OPA27](#) opamp from Texas Instruments. The opamp not only filters the digital signal from the PLL chip's charge-pump output, but also amplifies it to the 0.5 - 25V level required to make the most of the VCO's tuning range.

Following the VCO and resistive divider, a [GALI-5](#) MMIC from Mini-Circuits provides approximately 16 dB of gain. With the ROS-2150VW VCO and divider as shown, power available at the output jack at mid-band is approximately +14 dBm with less than +/- 2 dB of variation between 1000 and 1800 MHz.

Performance Considerations

A key specification of any PLL frequency synthesizer is the spectral purity of its output signal. Phase noise, also known as “jitter,” is caused by random short-term excursions in the carrier’s phase and can arise from a variety of causes. Amplitude noise is the other aspect of composite noise performance as observed with a receiver or spectrum analyzer, but it is usually insignificant compared to phase/frequency jitter. Finally, in addition to composite AM/PM noise, the synthesizer’s output signal may also contain discrete spurs that appear as sidebands on either or both sides of the carrier. For a detailed discussion of the nuances of noise and spur performance, see Dean Banerjee’s outstanding [PLL Performance, Simulation, and Design](http://www.national.com/appinfo/wireless/files/DeansBook_4_01.pdf), downloadable at http://www.national.com/appinfo/wireless/files/DeansBook_4_01.pdf. (A printed and bound version of Banerjee’s book is also available at <http://www.amazon.com>.)

Unlike a simple LC oscillator stage, the factors that can contribute to noisy or otherwise-impure signals at the output of a PLL are almost too numerous to count. Worse, it’s impossible to label a given synthesizer with a simple “spectral purity” specification that will allow direct comparisons with its peers. Noise and spurs are two-dimensional quantities – to evaluate their severity, we must consider both the amplitude relative to the carrier signal *and* the frequency offset from the carrier at which the effect in question was measured. Any synthesizer which will be used in a receiver or transmitter must be carefully designed to minimize noise and spurs at frequency offsets *which have the potential to degrade the receiver or transmitter’s performance*. The last point is significant - a synthesizer serving as a wideband FM receiver’s local oscillator can be designed to much looser specifications than one intended for use in a 40-meter contest-grade rig.

A detailed discussion of noise causes and cures is beyond the scope of this article. However, in a nutshell, PLL noise and spurious signal analysis must consider **reference noise**, **VCO noise**, and the **multiplication effect** of the loop. The design of the **loop filter** is critical, as the bandwidth of the loop should be tailored to the VCO’s needs. It is also important to consider extraneous noise-and-spur sources such as **power supplies** and **intermodulation effects** between various parts of the circuit.

Loop Multiplication Effect

Any phase-locked loop will increase the phase-noise amplitude of its reference source by $20 * \log(\mathbf{N})$ dB within the loop bandwidth, where \mathbf{N} is the loop's overall frequency-multiplication factor. Any discrete reference spurs within the loop bandwidth will also be amplified by a similar factor, noting that their frequency offset from the output carrier will remain the same as their offset from the reference. Consider a synthesizer with a 1-GHz output frequency and a 3-kHz loop bandwidth, whose reference frequency is 1 MHz with discrete sidebands at +/- 1 KHz and -80 dBc. \mathbf{N} is 1000 in this case, and $20 * \log(\mathbf{N})$ is 60 dB. At its output, the synthesizer will show sidebands at +/- 1 KHz and -20 dBc, which may not be acceptable for many applications. It is therefore critical to use a high-quality reference source for microwave PLLs with their high \mathbf{N} ratios -- or, failing that, a very narrow loop bandwidth with a high-quality VCO.

Reference-Related Noise and Spurs

The random phase-noise performance of a DDS is quite good. It is determined by either the noise of its own reference - typically a very clean crystal oscillator - or the noise-floor limitations of the VLSI process by which the DDS chip was fabricated (typically -140 dBc/Hz for ECL technology and -150 dBc/Hz or better for modern CMOS parts such as the AD9852). However, while its phase-noise performance is adequate for most purposes, the output of a DDS exhibits discrete sideband spurs due to phase truncation and timing tolerances in the lookup table and DAC. These are often significant enough to rule out the use of a standalone DDS as the first local oscillator in a high-performance HF receiver. Discrete spurs in the DDS reference can appear in the synthesizer's output signal at significant carrier offsets, falling away outside the loop bandwidth after being amplified by $20 * \log(\mathbf{N})$ dB within the loop bandwidth (see above).

In reality, our loop's vulnerability to spurs in its reference source is greater than one might expect. Both the National Semiconductor LMX2326 and Analog Devices ADF4112 PLL chips exhibit a readily-observed (yet apparently undocumented) tendency to respond to reference-

signal spurs at offset intervals corresponding to their internal comparison frequency, from DC to 100 MHz and beyond. Consider a loop with a 1 MHz comparison frequency, achieved by programming the PLL chip for a reference-divider modulus (**R**) of 10 with a DDS-generated reference signal at 10 MHz. As expected, DDS spurs close to the 10 MHz reference will appear in the synthesizer's output signal as noted above. Additionally, any DDS spurs appearing near 1 MHz intervals on either side of the 10 MHz reference frequency will appear in the output signal *exactly as if they had been generated in the vicinity of 10 MHz*. For example, a spur at 7.001 MHz would produce sidebands at +/- 1 KHz from the carrier at the output of the synthesizer. This effect has been noted with the National chip's reference/evaluation board as well as our prototype ADF4112-based synthesizer. It is clear that low-pass filtering the DDS reference is not enough – we must bandpass-filter it to suppress as many spurs as possible across the entire RF spectrum.

Our synthesizer addresses DDS spur problems by severely band-limiting the loop's reference signal with an inexpensive 4-pole monolithic crystal filter. With this 15-kHz-wide filter in place, a series of 5,000 automated measurements taken at randomly-selected frequencies between 1000 and 1800 MHz revealed no significant spurious responses at any frequency. Without the filter, the synthesizer's overall spur performance was dramatically worse. In the latter test, almost every randomly chosen frequency exhibited at least one noticeable spur.

The Loop Filter and VCO

The loop filter in a PLL is designed to attenuate high-frequency components of the loop error signal so that they do not modulate the VCO output. In practice, the loop filter cutoff is commonly set at between 2 and 5% of the reference frequency to obtain adequate attenuation of comparison-frequency sidebands appearing at the phase detector's output. In our circuit the comparison frequency is approximately 900 kHz, so this issue can be largely ignored. We consequently design the loop filter for best system phase-noise performance based on the VCO's own noise characteristics.

At offsets from the VCO carrier less than the loop filter bandwidth, the PLL attenuates noise contributed by the VCO. At offsets that are significantly greater than the PLL's loop

bandwidth, the principal source of noise is the VCO itself. Within the loop bandwidth, the noise performance is determined by the PLL reference source and the phase detector, and possibly the broadband noise floor of the frequency dividers as well. Even the noisiest VCO can be cleaned up by a PLL within its loop bandwidth, but extremely wide loop bandwidths carry a price of their own. As described in Analog Devices' data sheet for the [ADF4110/11/12/13](#), the inband noise floor of a typical inexpensive phase-frequency detector is limited to -85 to -90 dBc/Hz, after allowing for the multiplication effect of the loop as mentioned above. The phase-noise specifications of a good-quality VCO at offsets greater than 10 kHz from the carrier will be better than this -- typically around -95 to -110 dBc/Hz. Consequently, it makes sense to choose a loop bandwidth narrower than the carrier-offset points where the VCO's free-running noise profile crosses the PLL's own noise floor.

Besides requiring physically larger components, a narrower-than-necessary loop bandwidth allows more VCO noise to appear in the output signal. It also slows the loop lock time. The latter may be an issue where rapid tuning is required, such as in a sweep generator or spectrum analyzer application. Conversely, an excessively-wide loop bandwidth means that the PLL is actually contributing noise to the output, degrading the VCO's noise profile rather than improving it.

With the Mini-Circuits VCO family we've specified, loop bandwidths in the 1.5 kHz – 2.5 kHz range seem to yield the best compromise between component size, lock time, and noise performance. Some typical composite-noise results are shown in Figure 4. For comparison purposes, the red trace was obtained from the first local-oscillator section of an Icom IC-R7000 VHF/UHF communications receiver. The R7000's synthesizer splits its 770-1290 MHz output range between two separate VCOs, using a narrow loop bandwidth to suppress the reference-frequency spurs at 5 kHz. As a result, its close-in noise performance suffers relative to the hybrid synthesizer, but at offsets greater than a few kilohertz from the carrier, the Icom's dual discrete VCOs demonstrate their superiority.

There's no escaping the fundamental truth: a high-quality synthesizer design has to start with a high-quality VCO. As the Icom design shows, multiple oscillators with narrow tuning

ranges perform better than octave-band units like the ROS-2150VW, assuming the basic tank-circuit technology remains the same. Nevertheless, the Mini-Circuits parts hold up surprisingly well when compared to discrete-component VCO topologies like the IC-R7000's that are beyond the reach of most homebrewers.

Phase-Detector Noise and Comparison Frequency

Apart from the multiplication factor described above, noise contributed by the phase detector increases with increasing comparison frequency (the frequency at which the phase detector itself operates, after any prescaling and division is accounted for). Further, in our design, the higher the comparison frequency the wider the frequency range the DDS reference must cover, potentially making post-DDS filtering more difficult with off-the-shelf parts. Using a cheap 10.7 MHz FM crystal filter with 15 KHz bandwidth, we found a PLL reference frequency of about 1 MHz (10.7 MHz divided by 11) gave the best tradeoff between phase detector noise and DDS bandwidth.

Intermodulation Effects

Another potential source of discrete spurs in the synthesizer output is crosstalk between the DDS and PLL chips. The AD9852 DDS is a power-hungry device capable of radiating and conducting high-amplitude RF onto circuit-board traces in its vicinity. Without extensive physical shielding between the two components – e.g., placing them in separate RF-tight enclosures -- the PLL output exhibits spurs when tuned near a multiple of either 1/2 or 1/3 of the DDS chip's master clock frequency.

If the DDS is driven by a 100 MHz clock, for example, spurs can be observed on either side of the synthesizer's output signal when tuned near 1033 and 1050 MHz as well as most other multiples of 33 and 50 MHz. One spur appears at the fractional clock-frequency multiple in question, while an identical spur appears at the same offset on the other side of the output signal. Like other reference-derived spurs, these spurs begin to fall off in amplitude once their offset from the output frequency exceeds the PLL's loop bandwidth.

At a few of these problematic output frequencies, the crosstalk effect is severe enough to destabilize the loop and cause intermittent oscillation. The solution we've implemented takes advantage of the AD9852's built-in clock multiplier feature. Instead of clocking the DDS at a constant 100 MHz, we apply 10 MHz to the DDS's clock input and use its clock multiplier to select one of five possible clock frequencies between 80 and 120 MHz, maximizing the distance between any harmonic of $f_{DDSClk}/2$ or $f_{DDSClk}/3$ and the synthesizer's output frequency. This technique keeps the nearest fractional clock harmonic over 1.5 MHz away from the carrier at any given frequency, eliminating the problem entirely except for residual leakage of clock harmonics into the signal path associated with U204's input and output. Even without any additional shielding on the board, the latter spurs are seldom worse than -80 dBc.

While this approach avoids spurs due to this intermodulation effect, it does carry the penalty of increased software complexity. The extra calculations pose no significant burden to a PC or high-performance Atmel AVR controller, but may be a consideration if a less-capable microcontroller is used to drive the board.

Power supplies and noise

Some sections of the circuit are very sensitive to power supply noise. In particular, overall phase noise performance will be degraded if supplies to the VCO and the PLL chip are not adequately filtered and decoupled. Typical IC voltage regulators are very convenient to use, but they may exhibit wideband noise at magnitudes many times greater than a well-designed discrete-component regulator. We used Zener diodes as relatively-quiet references for critical parts of the circuit. Additionally, separate regulators are used to isolate different parts of the circuit, notably the digital and analog DDS sections.

Lock time

By toggling the synthesizer between its frequency extremes while monitoring the VCO tuning port with a digital oscilloscope at 5 milliseconds/division, the loop's lockup time and

damping characteristics can be observed directly. As the graphs in Figure 5 show, worst-case lockup time is approximately 25 milliseconds. Like the phase-noise graphs, the lock-time graphs were taken with an ROS-2150VW VCO, with the results compared to those obtained from the Icom IC-R7000's first local oscillator. The hybrid synthesizer's loop bandwidth was approximately 2 kHz.

Assembly and Operation

To obtain acceptable performance at microwave frequencies, it's necessary to use small components. The ICs used in the synthesizer come in TQFP, SOIC and SSOP packages with pin spacing as small as 0.6mm. It is nearly impossible to work with these devices without a printed circuit board. While our prototype was constructed dead-bug style on a bare copper-clad board, the author's eyesight and nerves have yet to recover from the experience!

Many hobbyists are intimidated by the precision and small dimensions involved with surface mount construction, but the truth is that SMT homebrewing is relatively easy to master with inexpensive tools, a modicum of patience, and a clever trick or two. In fact, surface-mount construction is a boon in disguise: because all of the components and their pins are accessible from the top of the board, a dual-layer surface-mount board carries almost all of the 'tweakability' advantages of dead-bug or Manhattan construction, while surface-mount components tend to be easier to purchase and stock due to industry standardization. Conversely, traditional DIP IC packages are fading from the scene at a frightening pace. Most modern RF ICs including DDS and PLL parts are simply not available in DIPs.

Soldering

Even if this is your first surface-mount project, you'll find that assembly goes smoothly with a few basic tools you may already own. A grounded soldering iron with a clean, fine tip is a must, and many constructors will find a lighted magnifier helpful as well. Manipulating the 1206-sized resistors and other smaller components is very difficult by hand, but a pair of cosmetology-grade tweezers such as the [3 3/4" Rubis](#) model

(http://www.folica.com/removal/rubis_costwe132.htm) makes the job trivial. Don't settle for whatever's on sale for \$1.99 at your local drugstore – when you're doing SMT work, a good pair of tweezers is one of those tools you won't want to live without. The Rubis tweezers have razor-sharp tips that can be used to straighten pins on the smallest SMT IC packages, as well as extract resistors from the tape strips they're embedded in.

Soldering the 0.6mm ICs – including the 80-pin DDS and the 16-pin PLL chip -- will likely be the most daunting part of the project, but it's not as difficult as you might expect. The secret? A roll of ordinary [solder-wick](#) (Digi-Key 50-4-25-ND). Clean the IC pads thoroughly with isopropyl alcohol prior to soldering – this goes for the rest of the board, too! – following up with a [liquid flux pen](#) (DigiKey KE1803-ND) if desired. After positioning and aligning each chip on the board, tack a couple of its corner pins into place with the soldering iron. Good pin alignment on all sides is critical, so it's important to keep the chip from shifting in place during soldering. Now, drag the soldering iron tip *lightly* across each row of pins while feeding fine-gauge solder to the point of contact. Take care not to apply enough force to bend any adjacent pins together – if this happens, fine-point tweezers or a dental pick will be necessary to get out of trouble.

You'll find that this process unavoidably creates numerous solder bridges along each row of IC pins. That's where the solder-wick comes in: simply lay a strip across the bridged pins and apply the iron to it. The result, believe it or not, will be a row of cleanly-soldered, unbridged pins that's almost indistinguishable from a professional wave- or reflow-soldering job. Don't be overly concerned about heat damage to the chips; use a hot (750F) iron and move along at a comfortable pace that gives the solder enough time to melt and flow around each pin. Heat soak caused by using a too-cold or dirty iron tip over the excessively-long period of time needed to make a good connection is a greater threat to the chip than momentary contact with a hot iron.

Most of the other components, including the SOIC ICs with their 0.050" pin spacing, do not need any special soldering techniques. However, it can be awkward to mount the resistors and capacitors due to the apparent need to use three hands to keep the part from moving. You'll find that there's usually enough solder on the pads to tack one end into place long enough to

solder the other end properly. Don't forget to return to the first pad and make the temporary connection permanent.

The VCO should be mounted with the dot or circle on its case to the upper right, towards C212. Note that its pin rows are offset slightly. Solder all 16 pads, again using a clean, hot iron to minimize heat soak. Get it right the first time, because the VCO, like the DDS chip, cannot be removed once soldered into place! Solder will flow under the part, out of the reach of solder-wick and into a place where it cannot be safely remelted without overheating the VCO or damaging the board.

Pay close attention to the bands on the electrolytic capacitors and Zener diodes. The polarity bands on the latter, D304 and D306, are difficult to see compared to the larger 1N4002 diodes used for reverse-polarity protection. Ideally, F401, the crystal filter, would be mounted with a spacer to keep its pads from shorting against the bottom of the filter's case. Since no spacer is provided by the manufacturer, you should leave a half-millimeter or so of clearance between the bottom of the filter and the PCB.

Finally, use isopropyl alcohol to clean up any excess flux remaining on the board after soldering. A Q-tip or sheet of optical-grade paper tissue is helpful here.

Enclosure and Shielding

The board is designed to fit into a Hammond 1590BB cast aluminum box. Applications in which the synthesizer board is not enclosed in its own chassis box may require extra shielding to suppress RFI from the DDS chip. Female SMA connectors are provided for the external clock input and synthesizer output signals. Power should be delivered through feedthrough capacitors, with the external data connection left to the builder's discretion, depending on what is driving the module. A DB-25 connector works well, allowing initial testing with PC-based software via the printer port. Table 1 describes the required connections between the board's 10-pin IDC connector and the PC- or Atmel AVR-based controller.

A total of 10 holes are provided in the board for 4-40 or 2-56 screws. When tightening these screws – and the mounting nuts for the SMA jacks -- use caution to avoid warping or

stressing the board excessively. Spacers will be needed beneath each screw hole since the bottom of a Hammond chassis box is not perfectly flat. SMT resistors and capacitors have almost no tolerance for physical stress, and components with cracks too small to see with the naked eye can be difficult to identify.

Power Consumption

Power is applied to the board via J501, a 4-pin header with separate pins for the VCO, DDS, and PLL supplies. Under normal circumstances, pins 2 and 3 should be tied together and connected to a clean, low-ripple +15V DC supply.

While pin 2 requires less than 100 mA, the current drain at pin 3 is influenced heavily by the AD9852's clock-multiplier setting. Applications which use the suggested internal clock rates of 80-120 MHz may need to supply as much as 400 mA to pin 3. Although we haven't encountered the phenomenon ourselves, a few AD9852/AD9854 users have observed incidents in which the chip has initialized itself in a state that causes it to draw excessive current when its reset pin (71) is not asserted during power-up. Consequently, we recommend adding a ½-amp fuse to the power-supply connection at pin 3 in applications that cannot guarantee a logic HIGH at DDSRESET (pin 2 of J501) at power-up time.

Pin 1 should receive either +15V or the maximum VCO tuning voltage required for the application, whichever is higher. Current drain at this pin is determined by the choice of VCO, and is typically less than 50 mA. Onboard regulation is provided for all supplies, but excessive input voltage at pin 3 should be avoided unless substantial heat-sinking is provided for U304. Don't run the board even for a few moments without any heat sinking at all at U302 and U304! It is normal for a fully-enclosed synthesizer assembly to run quite warm to the touch, especially at the higher DDS clock rates.

Reference Clock Sources

According to Analog Devices' data sheet for the [AD9852](#), the REFCLK input pin accepts either normal 3.3V CMOS logic levels for square-wave drive, or a 1V p-p sine wave centered

about 1.6V. R102 and R103 establish a 1.6-volt bias point for the REFCLK input and provide RF termination for a 10 MHz source with 50-ohm output impedance, with DC blocking by C102. The 1V p-p specification corresponds to approximately +4 dBm at J101. In practice, sine-wave clock signals at J101 from -10 dBm (200 mV pk-pk) to +10 dBm (2V pk-pk) appear to work well, with the DDS ceasing operation at -15 dBm.

Pads are provided on the board for U102, an optional 10 MHz clock module. When U102 is installed, R102 and R103 should be increased from 100 to 1,000 ohms, omitting C102 and J101. However, a high-quality ovenized (or at least temperature-compensated) external reference source is strongly recommended for most applications. At 1 GHz, a 50-ppm clock at U102 could put your signal 50 kHz off frequency!

Test and Demonstration Software

The synthesizer can be operated by a PC running Windows 95, 98, or ME, or by an Atmel microcontroller. This unique cross-platform capability is a consequence of our decision to use C++ to program the synthesizer, rather than Visual BASIC or other proprietary languages. While C++ is a more complex language than VB or Delphi, it's nevertheless possible to write highly-readable, portable code by eschewing the language's more arcane features.

C++ is supported by a broad range of compilers and development tools from many vendors. The Windows version of the control software is compiled with Microsoft's Visual C++, but the same code may also be compiled for the Atmel ATmega128 using the popular -- and free! -- [AVR-GCC](#) package (see notes at end of article for Web address). In fact, the GCC compiler is available on just about every platform under the sun including Windows and Linux, so even the Windows control program could be compiled under GCC with little effort. Instructions for compiling both the Windows MSVC and Atmel AVR-GCC versions are included in the PROJECT.MAK makefile, compatible with common MAKE utilities such as Microsoft's NMAKE. These commands may also be executed by themselves from a DOS session or batch file.

PC-hosted development environments can be handy for prototyping microcontroller-based projects. You don't have to continually program and reprogram your microcontroller to test new functionality, and debugging facilities offered by packages like Visual C++ are superior to anything available in the embedded world. Unfortunately, Windows NT-based operating systems such as Windows 2000 and Windows XP don't normally permit direct access to I/O ports from user-mode code, so our PC control software doesn't work on these platforms. (While workarounds do exist, they're beyond the scope of this article; check <http://www.geekhideout.com/iodll.shtml> for details.)

Windows Control Program

Two versions of the Windows control program, NSTEST.EXE and ASTEST.EXE, are provided. Both are simple Win32 console applications intended to be run from the DOS command line. If your synthesizer uses the National Semiconductor LMX2326 PLL chip from National Semiconductor, run NSTEST <n>, where <n> specifies the parallel port connected to the synthesizer board. For synthesizers constructed with the Analog Devices ADF4112 PLL chip, use ASTEST <n> instead. The differences between the two chips are not profound, but they do require slightly-different control code. By default, the control software expects to talk to a synthesizer based on the LMX2326 or ADF4112 chip, although the underlying SYNTH control class also supports the LMX2316 and LMX2306 parts as well as any ADF4112-compatible part in the ADF411X family.

Once the program is up and running, it should resemble the display in figure 6. Several options are provided to aid in testing and evaluation of the synthesizer. The 'R' key will tune the synthesizer to a random frequency between 1000 and 1800 MHz, while the 'E' key will allow the user to specify any desired frequency between 1000 and 2000 MHz (which may be entered directly in hertz or as a 3- or 4-digit value to be interpreted as an integral multiple of 1 MHz). The 0-8 keys will decrease the current frequency by 10^n Hz; holding down Shift while typing 0-8 will increase the current frequency by the same amount. The space bar will force the PLL and DDS

chips to be reprogrammed from scratch for the current frequency, which can be helpful if power to the synthesizer is interrupted while the program is running.

Both ASTEST.EXE and NSTEST.EXE are built from STEST.CPP, which contains several configuration options accessible only by modifying and recompiling the source code.

ANALYZER_CONTROL, which defaults to 0, can be set to 1 to enable the control program to tune a spectrum analyzer along with the synthesizer. This will also add options to the menu to acquire analyzer screenshots and execute automated random spur-searches, two powerful features that were used to help develop and document the project. If you happen to have a Tektronix 490P- or 2750P-series analyzer connected to your PC with a National Instruments GPIB interface, give the **ANALYZER_CONTROL** option a try!

WAIT_FOR_LOCK, which defaults to 1, causes the control software to wait for the PLL's lock-detect line to go active after any change in frequency. If the program displays a continuous string of periods after the frequency reading, it means that the loop is unlocked and troubleshooting is necessary. Recompiling STEST.CPP with **WAIT_FOR_LOCK** set to 0 will cause the program to ignore the lock-detect line.

FOUT_MIN, **FOUT_MAX**, **FOUT_RAND_MAX**, and **FOUT_DEFAULT** determine the minimum and maximum frequencies supported by the 'E' key; the maximum random frequency available by pressing 'R'; and the program's default power-up frequency, respectively. These may need to be changed if your synthesizer uses a VCO other than the ROS-2150VW.

DDS_CLOCK_FREQ should be set to the clock frequency provided at the DDS clock input jack (normally 10 MHz), while **DDS_CENTER_FREQ** should be left at 10.7 MHz for use with the crystal filter specified. **DDS_MIN_MULT** and **DDS_MAX_MULT** specify the range of possible DDS clock-multiplier constants, defaulting to 8 and 12 respectively.

Atmel ATmega128 Control Program

The ATmega128 test program, AVRSTEST.CPP, uses the same synthesizer control class declared in SYNTH.CPP as its PC-based counterpart. There the resemblance ends, however: AVRSTEST is a simple host program that displays a "logon" banner on an RS-232

terminal connected to the ATmega128's UART0 port, and waits for an incoming FRQ command to program the synthesizer to a given frequency. As with the PC program, the FRQ <n> command accepts values directly in hertz with 1-Hz precision or 3- or 4-digit values interpreted as integral multiples of 1 MHz.

The HyperTerminal program included with most versions of Windows offers an easy way to communicate with AVRSTEST (Figure 7). AVRSTEST assumes the ATmega128 chip is clocked at 16.0 MHz. Terminal settings should be 38,400 bps, no parity bits, 8 data bits, and 1 stop bit.

Other Atmel AVR processors are candidates for controlling the synthesizer, although not all offer the speed and comprehensive I/O capabilities of the ATmega128. Programming and construction details for the AVR family are beyond the scope of this article, but there are countless online resources that discuss part selection, support-circuit design, prototyping, and programming for the Atmel controllers, and the reader is encouraged to wade in with both feet. One excellent site is <http://www.avrfreaks.net/>, which enabled the author to climb the AVR learning curve in only a few days with no prior microcontroller experience.

The SYNTH Control Class

SYNTH.CPP contains all of the functionality needed to program the hybrid synthesizer. Although the SYNTH class was designed to be easily understood and reused, we'll examine it here in some detail for the benefit of readers who may not be familiar with the C++ language.

Creating and Initializing a SYNTH Object

To use SYNTH.CPP in your application, simply add it to your project's existing Win32 or Atmel C++ code with the #include directive. For each synthesizer module you wish to control, you'll need to declare an object instance of type **SYNTH**, passing various parameters to its C++ constructor:

```
SYNTH::SYNTH (SYNTH_PORT control_port,  
             PLL_CHIPTYPE chip_type,
```

```

S64      min_output_frequency,
S32      DDS_center_frequency,
S32      DDS_clock_frequency,
S8       DDS_min_clock_multiplier,
S8       DDS_max_clock_multiplier)

```

On the Atmel platform, **control_port** is a value of type **SYNTH_PORT** (**SYNTH_PORT_A** through **SYNTH_PORT_D**) that specifies which of four possible Atmel I/O ports is connected to the synthesizer hardware. On a Windows PC, **control_port** is the integer I/O address of the desired PC parallel port, typically 0x278, 0x378, or 0x3BC. (The PC test program STTEST.CPP determines the I/O address of the specified parallel port by reading the system's BIOS data area, another 'legacy' technique not supported by NT-based versions of Windows.)

chip_type is a value of type **PLL_CHIPTYPE** which corresponds to the PLL chip used by the synthesizer board. **LMX2306** specifies the National Semiconductor LMX2306, **LMX23X6** specifies the LMX2316 or LMX2326, and **ADF411X** is used to communicate with the Analog Devices ADF4110, ADF4111, ADF4112, or ADF4113.

min_output_frequency and **DDS_center_frequency** are the synthesizer's minimum supported output frequency and the post-DDS crystal filter's center frequency in Hz, respectively. These values help determine how the synthesizer fills in the gaps between PLL multiplier settings with the finely-tunable DDS reference.

DDS_clock_frequency specifies the frequency of the clock signal fed to the AD9852 in hertz, while **DDS_min_clock_multiplier** and **DDS_max_clock_multiplier** specify the permissible range of DDS clock-multiplier settings. Operating the synthesizer with **DDS_clock_frequency * DDS_max_clock_multiplier** products greater than 120 MHz is not recommended, since higher clock rates increase current consumption without offering any significant improvement in spectral purity. Internal clock rates greater than 200 MHz may cause damage to the DDS chip itself.

Below is an example of how to create a **SYNTH** object, taken from the Atmel test program AVRSTEST.CPP:

```

SYNTH synth(SYNTH_PORT_A, // Use AVR port A
            ADF411X,      // Talk to Analog Devices ADF4112 chip

```

```
97000000, // Minimum frequency 970 MHz for ROS-2150VW
10700000, // DDS center frequency = 10.7 MHz
10000000, // DDS clock input = 10 MHz
8, // Minimum DDS clock multiplier = 8X
12); // Maximum DDS clock multiplier = 12X
```

Using a SYNTH Object

Once you've created a **SYNTH** object, you can set the synthesizer's output frequency with the **SYNTH::set_frequency()** function and check for phase-lock acquisition with the **SYNTH::locked()** function. The **SYNTH::set_frequency()** function has the following C++ prototype:

```
void SYNTH::set_frequency(U64 hertz)
```

The **hertz** parameter is an unsigned 64-bit integer value that specifies the desired output frequency in hertz.

After tuning the synthesizer, you can call the **SYNTH::locked()** function to determine when the loop has locked at the new frequency:

```
BOOL SYNTH::locked(void)
```

Depending on the loop bandwidth, **SYNTH::locked()** will return TRUE within several milliseconds of a call to **SYNTH::set_frequency()**.

Below is a small code example showing how to tune the synthesizer object created above to a given frequency and wait for the command to succeed:

```
synth.set_frequency(1296001575); // Tune to 1296.001575 MHz
while (!synth.locked()); // Wait for phase-lock
```

Synthesizer Programming In Detail

You can create and use **SYNTH** objects in your own C++ code without knowing anything about programming the synthesizer hardware. For the curious, however, here are a few of the technical details underlying the **SYNTH** class. SYNTH.CPP is heavily-commented, but it does have aspects that may not be readily apparent.

Precalculation of DDS and Comparison-Frequency Constants

The **SYNTH** constructor's first task is to compute and store the **R** (reference) modulus for the PLL chip based on the **DDS_center_frequency** parameter. **R** determines the relationship between the 10.7 MHz reference frequency from the DDS and the actual PLL comparison frequency (**Fcomp**) where the phase detector operates. At reference frequencies under 16 MHz an **R** modulus of 11 is selected, yielding **Fcomp** ≈ 973 kHz for the standard 10.7 MHz reference. Higher reference frequencies, such as would be associated with a 21.4 MHz crystal filter, are divided by 32.

Once **R** is known, the following equation yields the required tuning range of the DDS synthesizer:

$$\text{DDS}_{\text{BW}} = \frac{\text{DDS_center_frequency}^2}{\text{min_output_frequency} * \text{R}}$$

For example, the 10.7-MHz DDS reference in a 1000 - 2000 MHz synthesizer will need to be tuned across a range approximately 10.4 kHz wide to cover the gaps between adjacent PLL **N** factors. This is a good match for the 15 kHz bandwidth of the ECS-10.7-15B crystal filter. It can be seen that higher DDS frequencies require broader crystal filters, leading to impaired spur performance.

After calculating the DDS tuning range, the minimum and maximum possible comparison frequencies are obtained by

$$F_{\text{comp}_{\text{min,max}}} = \frac{\text{DDS_center_frequency} \pm (\text{DDS}_{\text{BW}} / 2)}{R}$$

$F_{\text{comp}_{\text{min}}}$ is stored with the **SYNTH** object and used later by **SYNTH::set_frequency()** to determine the PLL's overall **N** division factor for the requested frequency. Needless to say, performing the $F_{\text{comp}_{\text{min}}}$ calculation when the **SYNTH** object is created rather than every time **SYNTH::set_frequency()** is called saves quite a bit of processing time!

Prescaler Modulus Selection

Most modern PLL chips, including those used by the synthesizer, contain dual-modulus prescalers that divide the sampled VCO signal prior to applying it to the A and B counters that perform the loop's divide-by-**N** function. Because of the way these three counters -- **A**, **B**, and **P** - interact, the dual prescaler factors **P** and **P+1** determine the minimum overall **N** value the chip can support⁸ according to the equation $N_{\text{min}} = P^2 - P$.

The National Semiconductor chips have fixed prescaler constants that the software must look up based on the **chip_type** parameter. The LMX2306, intended for lower-frequency use through 550 MHz, offers a relatively-small **P** factor of 8/9 corresponding to $N_{\text{min}} = 56$. The higher-frequency LMX2316 and LMX2326 parts use **P** constants of 32/33, so their minimum **N** factor is substantially higher ($N_{\text{min}} = 992$). Note that when $F_{\text{comp}_{\text{max}}} = 973$ kHz as determined above, the LMX2316/26 parts are incapable of operation below 965 MHz (973 kHz * 992)! To use the LMX2316/26 parts at lower frequencies, the **R** modulus calculation will need to be modified to yield a lower $F_{\text{comp}_{\text{max}}}$, possibly degrading phase-detector performance in the process.

The Analog Devices ADF411X chips offer a more flexible alternative by supporting programmable **P** factors from 8/9 to 64/65. When **chip_type** is set to **ADF411X**, the **SYNTH** constructor calculates N_{min} based on $F_{\text{comp}_{\text{max}}}$ and the specified **min_output_frequency**, and selects the largest **P** factor that will satisfy the $N \geq P^2 - P$ relation.

PLL Initialization

Once the **P** and **R** modulus values have been determined, the PLL chip is ready for initialization. Again, this is done only once when the **SYNTH** object is constructed, rather than every time a new output frequency is requested. The chip's digital lock-detection feature is also enabled at this point. The initialization procedure is slightly different for the National and Analog parts, due to the latter's programmable prescaler and charge-pump current control features.

DDS Initialization

Finally, the **SYNTH** constructor must reset and initialize the AD9852 DDS. This step has been a source of confusion to many AD9852 users, because the chip powers up in a mode that causes its internal registers to update themselves periodically whether the initialization data has been completely transmitted or not! Our approach is to initialize the DDS chip twice with identical command words, turning off the auto-update feature and setting the minimum clock-multiplier factor by default. Portions of the chip that are unused by the synthesizer are turned off to save power, including the secondary or quadrature DAC and internal comparator (which is replaced by the LT1016 in our circuit due to its superior sensitivity and output level). Even if the first initialization attempt is incomplete due to an asynchronous register update, the second attempt will ensure that the entire initialization command is latched manually at the proper time.

SYNTH::set_frequency()

When the synthesizer is tuned to a new frequency, the **SYNTH::set_frequency()** function first calculates the PLL's overall **N** division factor by dividing the output frequency by the previously-calculated **F_{comp_min}**. The required DDS reference frequency **D** is then given simply by

$$D = F_{out} * R / N$$

Because the **D** value is computed in 56.8 fixed-point format, the DDS reference frequency can be set with 1/256 Hz precision, corresponding to tuning steps roughly 0.73 Hz wide at a 2 GHz output frequency. (A 76 GHz rig can be tuned in 30 Hz steps with this degree of control -- so given the wide availability of < 1-ppb crystal, GPS, and rubidium standards on eBay, there's no excuse for not knowing where you are!)

Next, the PLL's **A** and **B** counters are programmed with the calculated **N** value, and a series of `assert()` statements ensure that the resulting values make sense for the PLL chip type. Attempting to tune to an out-of-range frequency will yield a fatal error in the Windows test program. Phase continuity during most incremental tuning operations is ensured by skipping the PLL programming step if the **N** value has not changed since the last call to **SYNTH::set_frequency()**. This way, the synthesizer can be tuned smoothly with a rotary encoder without generating clicks or other objectionable artifacts in received audio.

The AD9852 is programmed to generate the reference frequency **D** in two steps. First, the clock multiplier **CM** is chosen according to the algorithm described in "Intermodulation Effects" above. Because a change to the clock multiplier reinitializes the DDS chip, requiring extra processing time and breaking phase continuity, the clock multiplier is not reprogrammed if the previous **CM** value is still optimal for the new output frequency. After the clock-multiplier calculation step, the quotient of the **D** value and (**DDS_clock_frequency** * **CM**) is written to the DDS phase-increment register at 36-bit precision, and the loop begins its phase-locking process.

SYNTH::locked()

The **SYNTH::locked()** function returns the logic level at pin 14 of the PLL chip. This is a digital lock-detection output that goes high after the detected phase error remains below a predetermined minimum for a period of five cycles of **F_{comp}**.

Schematic Notes

Power Supply (Figure 8)

D306 determines the voltage supplied to the VCO module, minus the voltage drop across the series pass transistor of approximately 0.6V. See the Parts List for recommended D306 part numbers for 5V and 12V VCOs.

PLL and VCO (Figure 9)

Loop-filter components C212, C213, C216, R208, and R209 are determined according to the equations from the Banerjee reference (see sidebar).

If an Analog Devices ADF4110, ADF4111, ADF4112, or ADF4113 is selected for U201, R217 must be installed between pin 1 and ground to bias the chip's internal charge-pump current source. Boards assembled with the National Semiconductor LMX2306, LMX2316, or LMX2326 should omit R217.

C230 is included to discourage any tendency on the PLL chip's part to lock to VCO harmonics. During testing, the ROS-2150VW VCO's second-harmonic level was observed to increase by as much as 2-3 dB relative to a 1 GHz fundamental at temperatures close to 0 degrees F, causing occasional lock failures when the board was initially powered up below room temperature. A 1-pF capacitor between the differential FIN pins prevented this symptom from occurring. Even-larger shunt capacitor values may be necessary with other VCOs, especially when the synthesizer is operated well below the upper frequency limit of the PLL chip.

Similarly, when working with other VCOs, the divider network consisting of R210-R212 and R218 may need to be adjusted to keep the sampled VCO signal at pin 6 of U201 within the manufacturer's recommended 0 to -10 dBm range.

DDS and Clock (Figure 10)

Provision is made for an internal clock module to be installed at U102, although higher-quality external clock sources are recommended. If an onboard clock source is installed, the components associated with the external-clock input (R102, R103, C102, J101) should be omitted.

U101 may be either an Analog Devices AD9852AST or AD9854AST. The AD9854 supports the generation of in-phase and quadrature signals -- a feature not used by the synthesizer -- but is otherwise identical to the AD9852. The -AST suffix refers to the family of parts designed for a maximum internal clock rate of 200 MHz. While the more-expensive -ASQ part supports clock rates up to 300 MHz, its conductive bottom surface must be reflow-soldered to the PCB to dissipate enough heat for operation beyond 200 MHz.

Post-DDS Filtering (Figure 11)

F401 is designed to be terminated with 3,000 ohms for minimum passband ripple, but it works well in this non-critical application with 36:1 broadband transformers and a 50-ohm resistive termination at U401. The low-pass filter consisting of L402, L403, and C403-405 helps reject DDS image signals at higher frequencies where F401's parasitic elements degrade its attenuation.

Digital Control Interface (Figure 12)

R501 and R507-509 help protect U201 from ESD and transients, while the dividers formed by R501-506 and R510-514 allow 5V logic families to drive the 3.3V DDS chip (U101)'s data lines safely. Table 1 details the connections between J501 and the IBM PC parallel port or Atmel AVR microprocessor used to control the synthesizer.

Analog Devices recommends that the DDSRESET line (J501 pin 2) be asserted HIGH during power-up initialization. When using the PC parallel port, it's a good idea to include a ½-

amp fuse in the power-supply line to J301 pin 3 to protect the chip against invalid initialization states.

Complete Parts List Including [Digi-Key](#), [Mini-Circuits](#), and [Analog Devices](#)

Part Numbers

R101: 1.3K ohms ([P1.3KECT-ND](#))

R102, 103: 100 ohms ([P100ECT-ND](#)) (50-ohm external clock input only)

R102, 103: 1K ohms ([P1.0KECT-ND](#)) (Internal clock source only)

R403: 1K ohms ([P1.0KECT-ND](#))

R104: 3.9K ohms ([P3.9KECT-ND](#))

R105-108, 212: 51 ohms ([P51ECT-ND](#))

R201-203: 18 ohms ([P18ECT-ND](#))

R206, 207, 510-514: 10K ohms ([P10KECT-ND](#))

R210, 211, 218: 22 ohms ([P22ECT-ND](#))

R213-216: 470 ohms ([P470ECT-ND](#))

R217: 4.7K ohms ([P4.7KECT-ND](#)) (ADF4112 PLL chip only; omit for LMX2326)

R301, 303: 4.7K ohms ([P4.7KECT-ND](#))

R401, 402, 501, 507-509, 109-112: 100 ohms ([P100ECT-ND](#))

R502-506: 3.3K ohms ([P3.3KECT-ND](#))

C101, 103, 207, 208, 209, 401: 0.01 uF/50V ceramic ([399-1234-1-ND](#))

C102, 104, 106, 202, 204, 205, 211, 215, 219, 224, 226, 309, 311, 312, 315, 317, 318, 407, 408:
0.1 uF/50V ceramic ([PCC104BCT-ND](#))

C201, 203, 206, 210, 217, 220: 100 pF/50V ceramic ([399-1205-1-ND](#))

C214, 321: 47 uF/35V electrolytic ([PCE3280CT-ND](#))

C218, 228, 302, 304, 307, 308, 313, 314, 409: 10 uF/16V tantalum ([399-1595-1-ND](#))

C221-223, 225, 406: 0.001 uF/50V ceramic ([311-1170-1-ND](#))

C230: 1 pF/50V ceramic (see text) ([399-1178-1-ND](#))

C301, 319, 320, 322: 10 uF/35V electrolytic ([PCE3413CT-ND](#))

C403, 405: 270 pF/50V NPO ceramic ([399-1209-1-ND](#))

C404: 470 pF/50V NPO ceramic ([399-1213-1-ND](#))

D301-303: 1N4002 ([DL4002DICT-ND](#))

D304: 5.6V 500 mW Zener ([BZT52C5V6-7DICT-ND](#))

D306: 5.6V 500 mW Zener (5V VCOs only) ([BZT52C5V6-7DICT-ND](#))

D306: 12V 500 mW Zener (12V VCOs only) ([BZT52C12-7DICT-ND](#))

F401: ECS-10.7-15B monolithic crystal filter, +/- 7.5 kHz bandwidth, 25 kHz stopband ([X704-ND](#))

J101, 201: SMA female bulkhead jack ([J569-ND](#))

J301: 4-position straight header ([A1912-ND](#))

J501: 10-position shrouded header ([MHB10K-ND](#))

L402,403: 0.91 uH RF choke ([DN1015CT-ND](#))

L201, 202, 204-207, 301, 404: 33 uH 115 mA RF choke ([TKS2638CT-ND](#))

L203: Mini-Circuits [ADCH-80A](#)

Q301, 303: 2SC1847 ([2SC18470Q-ND](#))

T401, 402: Mini-Circuits [T36-1-KK81](#) 36:1 RF transformer

U101: Analog Devices [AD9852AST](#) DDS synthesizer

U102: ECS-3953M 10 MHz oscillator module for internal-clock option ([XC288CT-ND](#))

U201: Analog Devices [ADF4112BRU](#) 3.0 GHz RF PLL frequency synthesizer

U201: National Semiconductor [LMX2326TM](#) 2.8 GHz RF PLL frequency synthesizer ([LMX2326TM-ND](#))

U202: Burr-Brown OPA27 low-noise operational amplifier ([OPA27GU-ND](#))

U203: Mini-Circuits [ROS-2150VW](#) 1-2 GHz VCO

U204: Mini-Circuits [GALI-5](#) monolithic amplifier

U301, 302: Linear Technology LT1086-3.3 voltage regulator ([LT1086CT-3.3-ND](#))

U304: 7812 voltage regulator ([NJM7812FA-ND](#))

U401: Linear Technology LT1016 precision comparator ([LT1016CS8-ND](#))

Miscellaneous hardware and optional parts

Hammond 1590BB aluminum enclosure ([HM152-ND](#))

TO-220 mounting kits for U301, U302 ([4724K-ND](#))

10-position IDC socket for J501 ([MKC10A-ND](#))

10-position ribbon cable ([MC10G-5-ND](#))

Loop filter components (ROS-2150VW VCO, 2.5 kHz loop bandwidth):

R208: 150 ohms ([P150ECT-ND](#))

R209: 82 ohms ([P82ECT-ND](#))

C212: 1 uF/35V tantalum ([PCS6105CT-ND](#))

C213: 0.1 uF/50V ceramic ([PCC104BCT-ND](#))

C216: 0.1 uF/50V ceramic ([PCC104BCT-ND](#))

PC Boards

Dual-layer PC boards with plated vias, solder mask, and component markings are available from John Miles, KE5FX (jmiles@pop.net) for US \$25.00 postpaid. Cash, checks, and PayPal are accepted.

Postscript: Collaborative Design in the Internet Age

This board was designed entirely via email -- the two authors have never met! [Richard, VK6BRO](#) was courageous enough to devote substantial personal labor to the project sight-unseen, fashioning the schematics and PC board layout from a few vague sentences from [John, KE5FX](#) ("Pin 10 goes to pin 15, with a 33 uH RF choke to the supply bus.") Despite the 9,000 miles that separated our QTHs, the design process could scarcely have been easier with two engineers in adjacent cubicles. Our synthesizer boards have undergone substantial testing over the past year, and we're both eager to hear the experiences of others who've duplicated our work or improved upon it.

Additional thanks go to Kathy Stenger for her drafting assistance.

References

1. QUALCOMM Inc., "Hybrid PLL/DDS Frequency Synthesizers," AN2334-4 (CL80-3459-1A), March 1992
2. U.S. Patent No. 4,965,533 *Direct Digital Synthesizer Driven Phase Lock Loop Frequency Synthesizer*, QUALCOMM, Inc.
3. U.S. Patent No. 5,028,887 *Direct Digital Synthesizer Driven Phase Lock Loop Frequency Synthesizer with Hard Limiter*, QUALCOMM, Inc.
4. U. L. Rohde DJ2LR / KA2WEU, "A High Performance Hybrid Synthesizer," QST, March 1995, pp 30-38
5. Rohde, *Microwave and Wireless Synthesizers: Theory and Design* (New York: John Wiley and Sons, ISBN 0471520195, 1997), pp 489-504
6. C. Drentea KW7CD, "Beyond Fractional N," QEX, March/April 2001 pp 18-25 and May/June 2001 pp 3-9
7. D. Banerjee, *PLL Performance, Simulation, and Design, 2nd Edition* (National Semiconductor Corporation, ISBN 0970820704, 2001), pp. 105-113
8. Banerjee, p. 160

9. R. Witte, *Spectrum and Network Measurements* (Norcross, Georgia: Noble Publishing, ISBN 1884932169, 2001), pp 147-148

Software and Supplementary Files

<http://www.qsl.net/ke5fx/synth.html>

Data Sheets and Other Online Resources

[AD9852](http://products.analog.com/products/info.asp?product=AD9852): <http://products.analog.com/products/info.asp?product=AD9852>

[LMX2306/16/26](http://www.national.com/pf/LM/LMX2326.html): <http://www.national.com/pf/LM/LMX2326.html>

[ADF4110/11/12/13](http://products.analog.com/products/info.asp?product=ADF4112): <http://products.analog.com/products/info.asp?product=ADF4112>

[GALI-5](http://www.minicircuits.com/cgi-bin/spec?cat=amplifier&model=GALI-5): <http://www.minicircuits.com/cgi-bin/spec?cat=amplifier&model=GALI-5>

[ADCH80A](http://www.minicircuits.com/dg03-242.pdf): <http://www.minicircuits.com/dg03-242.pdf>

[LT1016](http://www.linear.com/prod/datasheet.html?datasheet=157): <http://www.linear.com/prod/datasheet.html?datasheet=157>

[OPA27](http://focus.ti.com/docs/prod/folders/print/opa27.html): <http://focus.ti.com/docs/prod/folders/print/opa27.html>

[ROS-2150VW](http://www.minicircuits.com/cgi-bin/spec?cat=vco&model=ROS-2150VW): <http://www.minicircuits.com/cgi-bin/spec?cat=vco&model=ROS-2150VW>

[National Semiconductor's WEBENCH \(an excellent online PLL modelling program\)](http://wireless.national.com/)
<http://wireless.national.com/>

[Analog Devices ADIsimPLL Version 2.0](http://forms.analog.com/Form_Pages/RFComms/ADIsimPLL.asp)
http://forms.analog.com/Form_Pages/RFComms/ADIsimPLL.asp

[Mini-Circuits application notes for VCOs and other components](http://www.minicircuits.com/application.html)
<http://www.minicircuits.com/application.html>

[AVR-GCC \(C++ compiler for Atmel microcontrollers\)](http://www.avrfreaks.net/AVRGCC/)
<http://www.avrfreaks.net/AVRGCC/>

Appendix (Sidebar) 1: Spectral Noise Measurement

Spur and noise amplitudes are expressed relative to a desired carrier signal. A spur at –80 dBc has an amplitude 80 dB below that of the carrier. Unlike CW spur signal power, though, noise power is proportional to the square of the bandwidth over which it is measured; noise levels measured by a receiver with a 1-kHz IF filter cannot be directly compared to levels measured with a 100-Hz filter. So noise power is usually expressed in units of *dBc/Hz*, i.e., dB below the carrier amplitude, normalized to a bandwidth of 1 Hz. When using a spectrum analyzer to measure composite noise, its resolution bandwidth setting (which corresponds to the IF filter bandwidth in a traditional receiver) can range from 1 Hz to 1 MHz or more. As a first approximation, the noise level observed on a receiver or spectrum analyzer at a given bandwidth can be normalized to 1 Hz by subtracting $10 * \log_{10}(\text{BW})$.

However, this isn't the whole story. The shape factor of a filter affects its *noise equivalent bandwidth*, which is another way of saying that a filter's 3-dB points measured or modelled with a CW signal may not accurately reflect its response to broadband noise. A spectrum-analyzer filter designed with a gradual slope to reduce ringing and distortion during fast sweeps may exhibit a noise equivalent bandwidth substantially wider than its specifications suggest. Finally, the logarithmic amplifier and video detector stages may also introduce errors in the analyzer's apparent response to noise. For these reasons, most traditional (non-FFT-based) spectrum analyzers require a correction factor of +2 to +2.5 dB to be added to normalized noise powers.⁹

Many newer analyzers include a "noise normalization" feature that computes the $\log(\text{BW})$ and assorted correction factors automatically. The Tektronix 494AP used to capture the spectral-purity graphs in this article displays normalized noise values in dBm/Hz, so all of the graphs were recorded with a –16 dBm offset value to compensate for the +16 dBm amplitude of the 1 GHz carrier being measured.

Appendix (Sidebar) 2: Loop Filter Component Selection

When choosing a particular VCO or loop bandwidth for your synthesizer, it will be necessary to recalculate the loop-filter component values. This is a step-by-step process that begins with a few key loop parameters.

K_{vco} is the VCO's tuning-port sensitivity figure in Hz/volt. This may vary by a factor of 2:1 or more over the VCO's tuning range; varactor-tuned VCOs are less sensitive at the upper end of their range. In our calculations with the ROS-2150VW we use its K_{vco} figure at 1000 MHz (70 MHz/volt), but loop performance can be further optimized by looking up – or experimentally measuring -- K_{vco} at the frequency of interest.

K_{pfd} is the phase detector's current gain in amperes per 2π radians of phase error. K_{pfd} is not well-characterized by many manufacturers; some specify it in mA per radian, while others neglect to mention whether or not it's dependent on the PLL chip's supply voltage. Analog Devices employs a constant-voltage source with an external resistor to establish their chips' maximum K_{pfd} figure, while the LMX2326's figure is 1 mA for a 3V supply and "is about 25% more when operating at 5 Volts versus 3 volts and varies with voltage in a linear fashion," according to the online help at National Semiconductor's [WEBBENCH](#) page (see notes at end of article for Web address). For our example, we'll select a programmed value of 1.88 mA for the ADF4112.

F_{out} is the output frequency in Hz. For synthesizers with wide tuning ranges, we take the geometric mean (the square root of the product) of its maximum and minimum frequencies. To cover 1000 to 2000 MHz, for example, F_{out} would be 1414 MHz.

F_{comp} is the comparison frequency in Hz. We assume the DDS reference is centered at 10.7 MHz, and choose an R divisor of 11 to yield $F_{comp} = 973$ kHz.

F_c is the loop bandwidth in Hz. We'll use 2500 Hz in our example calculation, since this yields readily-available component values and results in fast lock times without degrading the VCO's noise performance.

ϕ is the desired phase margin, typically between 40 and 55 degrees. 48 degrees is a good compromise between minimal spectral "shoulders" and lock time.

T_{31} is the ratio of the time constants of poles T3 (R209, C216) and T1 (R208, C212, and C213) in the active-filter topology we've selected. Banerjee's suggested T_{31} constant of 0.5 is used.

C_3 is the capacitor value in farads at the final pole before the VCO, C216 in our case. Values of a few hundred pF are common, but these lead to large (read: noisy) series resistor values. In the author's experience, as long as the op-amp is stable driving a capacitive load, relatively-large C_3 values have the benefit of keeping wideband noise from the rest of the loop out of the VCO. The OPA27 used in our loop works well with $C_{216} = 0.1 \mu\text{F}$.

First, the phase margin and loop bandwidth are converted to radian-based notation:

$$\phi = \phi * 2 * \pi / 360.0$$

$$\omega_c = 2 * \pi * f_c$$

The time constants of the three filter poles can then be determined:

$$T_1 = ((1.0 / \cos(\phi)) - \tan(\phi)) / \omega_c * (1.0 / (1.0 + T_{31}))$$

$$T_3 = T_1 * T_{31}$$

$$T_2 = 1.0 / (\omega_c^2 * (T_1 + T_3))$$

Next, the loop's **N** factor (or its geometric mean, in this case) is calculated along with a few intermediate results:

$$N = F_{out} / F_{comp}$$

$$C_{x_a} = (K_{pfd} * K_{vco}) / (\omega_c * \omega_c * N)$$

$$C_{x_b} = (1.0 + (\omega_c^2 * T2^2)) / ((1.0 + (\omega_c^2 * T1^2)) * (1.0 + (\omega_c^2 * T3^2)))$$

$$C_x = C_{x_a} * C_{x_b}^{0.5}$$

The loop-filter component values are now available, with the results for our example 2500 Hz loop filter shown in parentheses:

$$C_{212_{\mu F}} = 1E6 * C_x \quad (0.98 \mu F)$$

$$C_{213_{\mu F}} = 1E6 * C_x * (T1 / (T2 - T1)) \quad (0.11 \mu F)$$

$$C_{216_{\mu F}} = 1E6 * C3 \quad (0.1 \mu F)$$

$$R_{208_{\Omega}} = 1E6 * T2 / (C_{212} + C_{213}) \quad (152 \text{ ohms})$$

$$R_{209_{\Omega}} = 1E6 * T3 / C_{216} \quad (81 \text{ ohms})$$

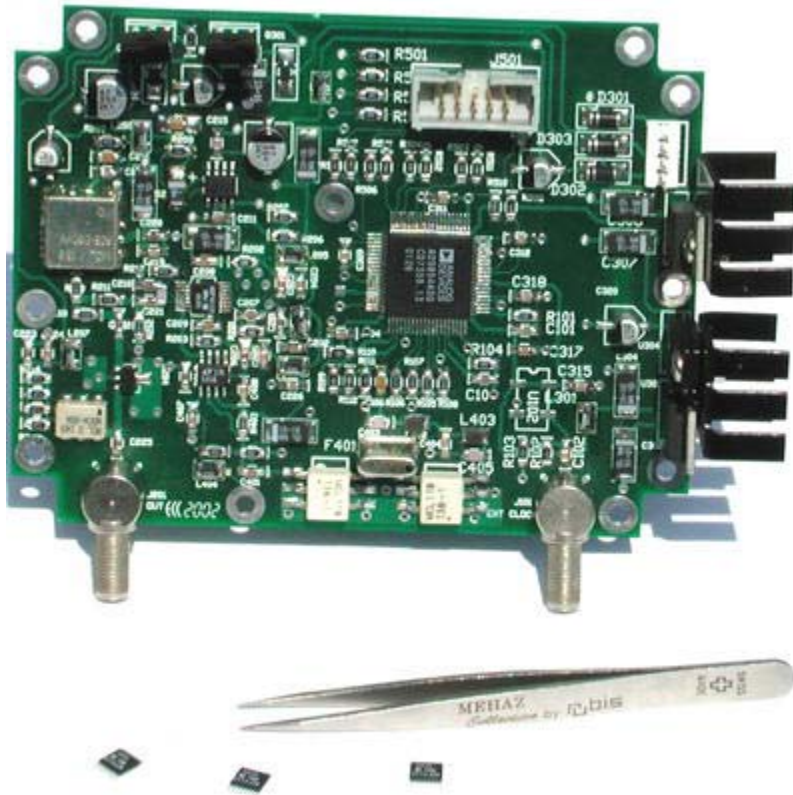


Fig 1 Complete synthesizer PC board with output and external-clock jacks

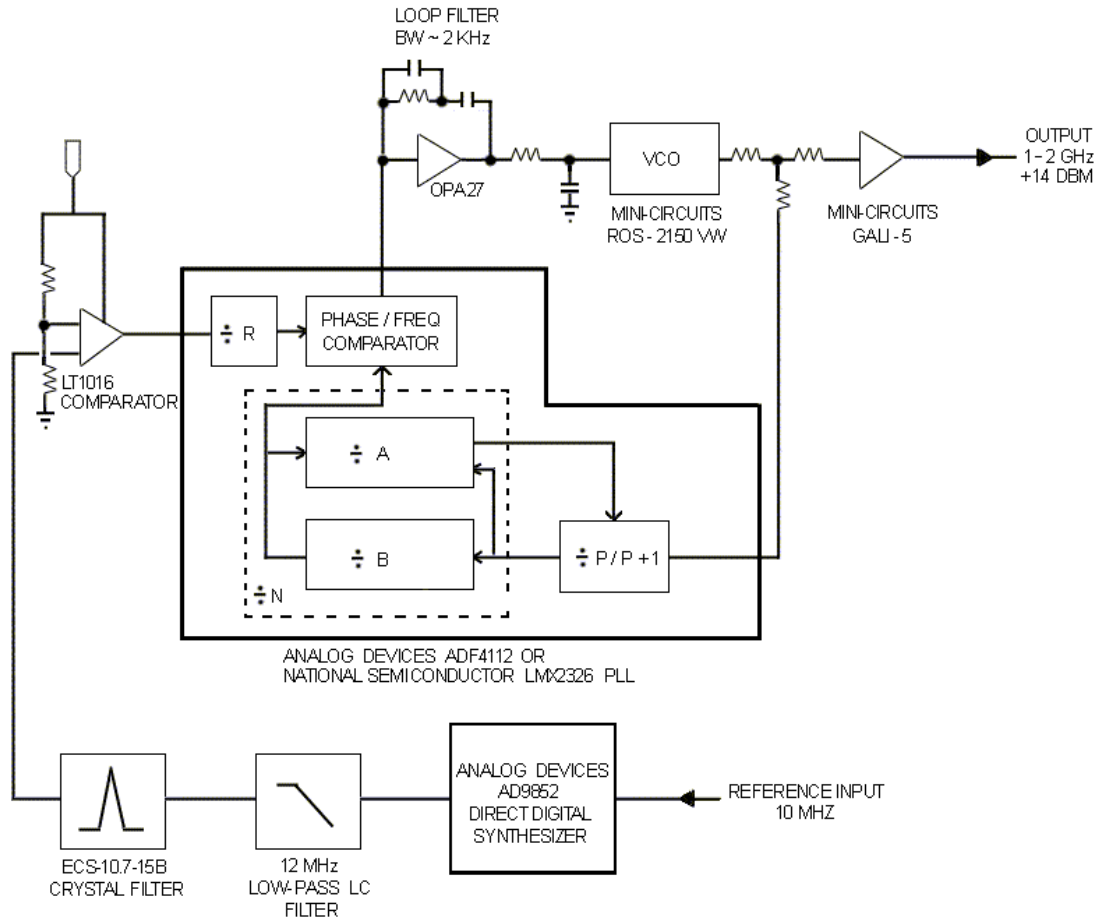


Fig 2 Block diagram

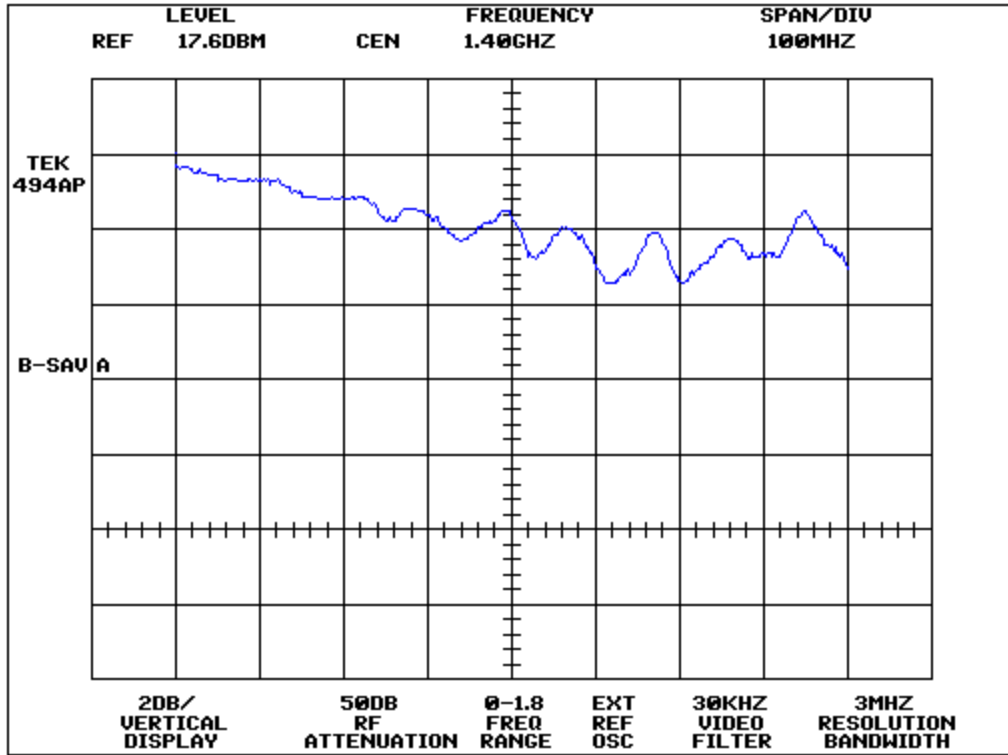
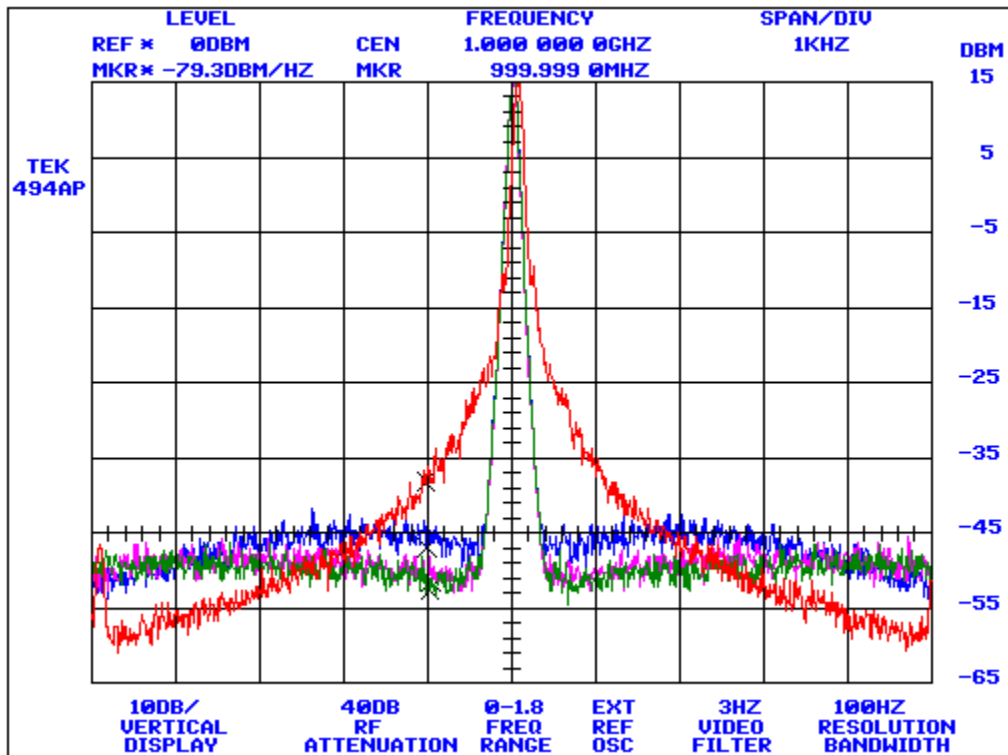
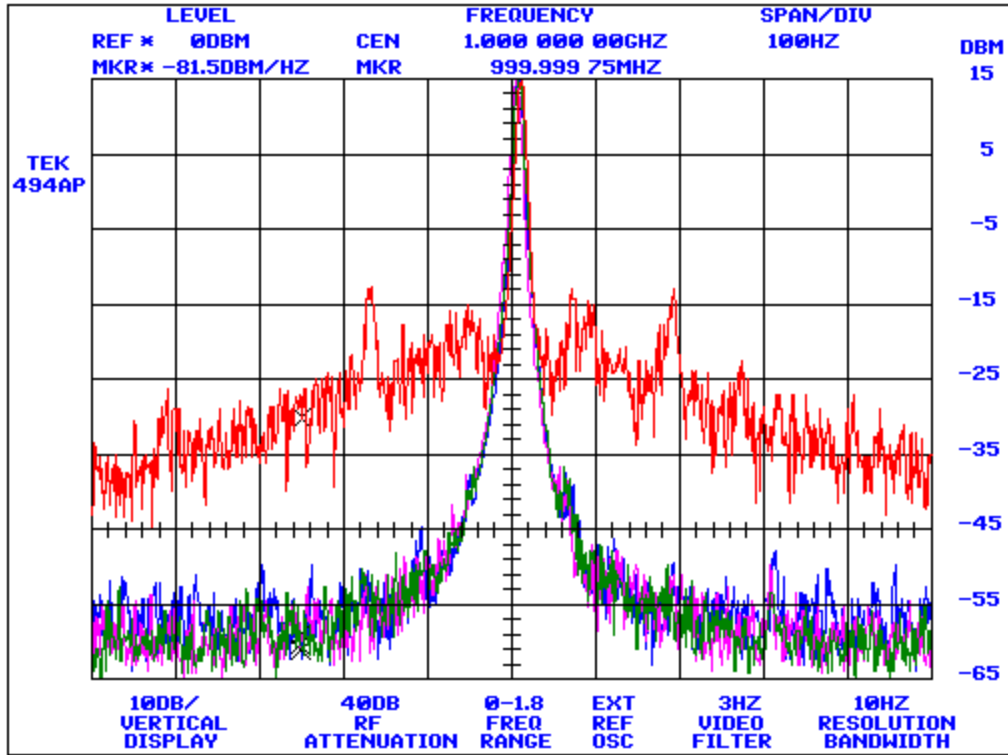


Fig 3 Output flatness from 1000 to 1800 MHz at 2 dB/vertical division



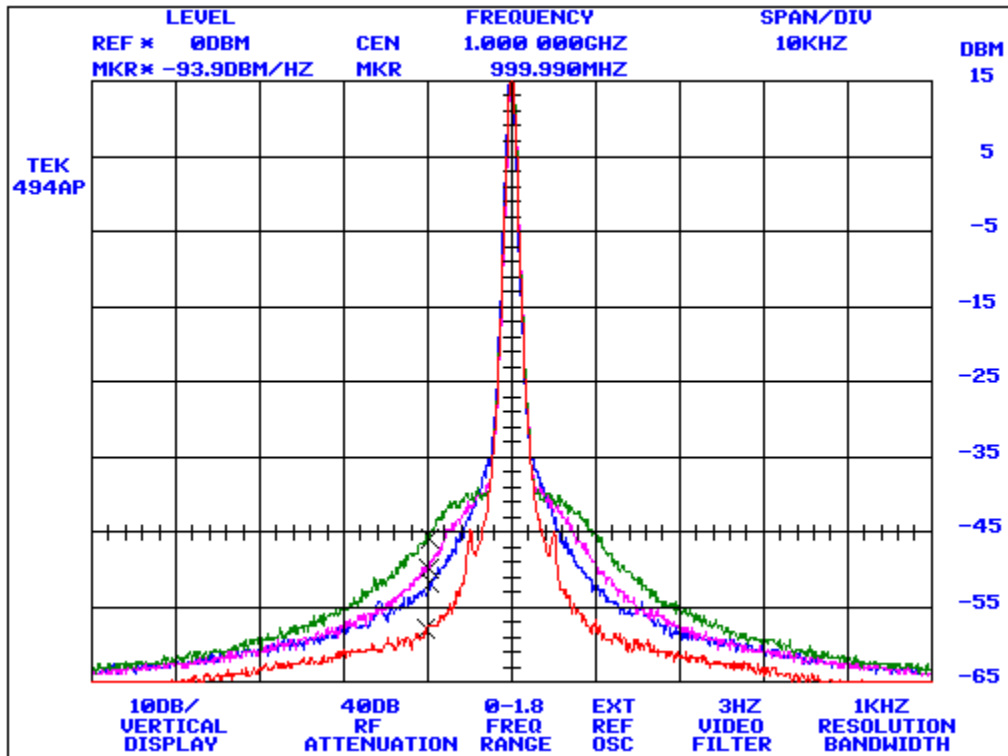
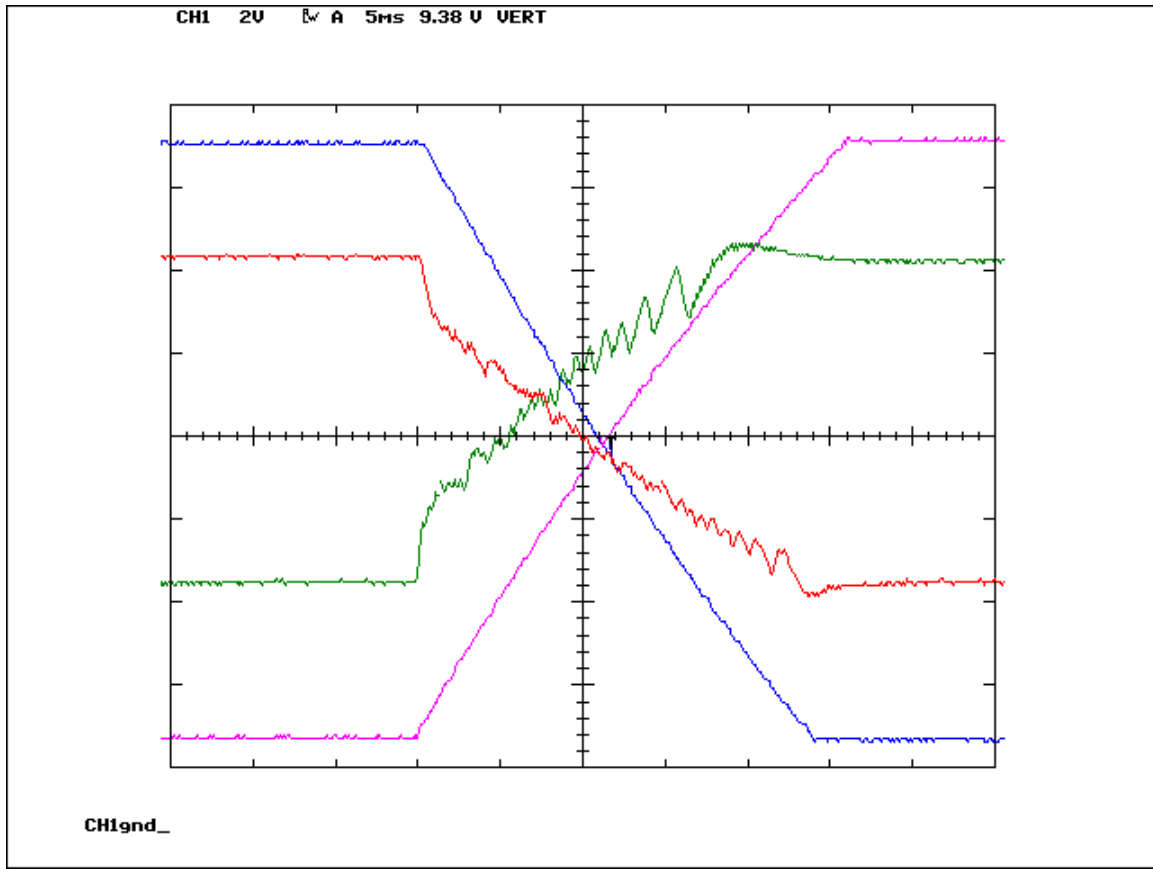
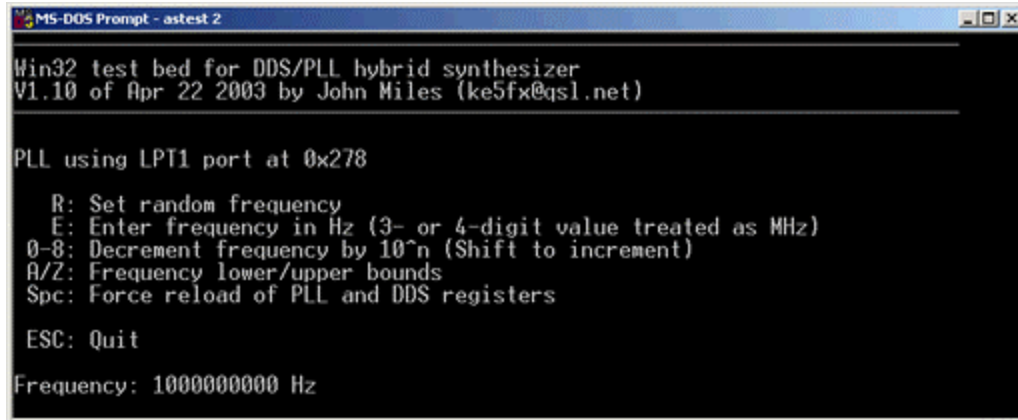


Fig 4 Composite noise performance. The blue trace corresponds to the synthesizer's composite-noise profile at a loop-filter bandwidth of 1,500 Hz; the purple trace corresponds to 2,500 Hz; and the green trace to 3,500 Hz



Icom IC-R7000 first LO synthesizer VCO tuning line, 770 MHz to 1030 MHz
Icom IC-R7000 first LO synthesizer VCO tuning line, 1030 MHz to 770 MHz
KE5FX hybrid synthesizer VCO tuning line, 1000 MHz to 2000 MHz
KE5FX hybrid synthesizer VCO tuning line, 2000 MHz to 1000 MHz

Fig 5 Synthesizer lockup characteristics compared to Icom IC-R7000



```
MS-DOS Prompt - astest 2
Win32 test bed for DDS/PLL hybrid synthesizer
V1.10 of Apr 22 2003 by John Miles (ke5fx@qsl.net)

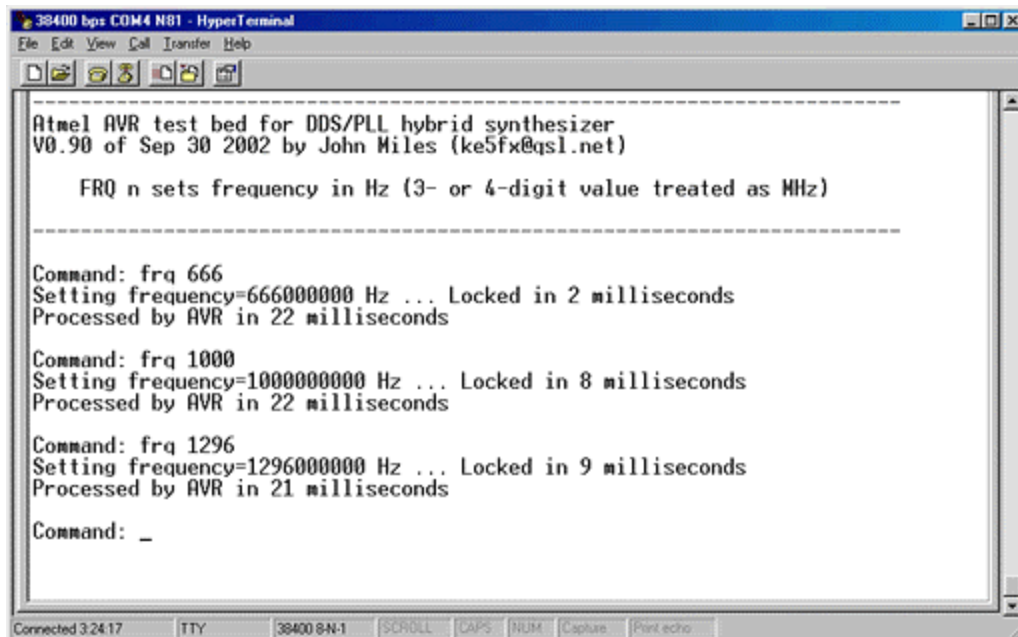
PLL using LPT1 port at 0x278

R: Set random frequency
E: Enter frequency in Hz (3- or 4-digit value treated as MHz)
0-8: Decrement frequency by 10^n (Shift to increment)
A/Z: Frequency lower/upper bounds
Spc: Force reload of PLL and DDS registers

ESC: Quit

Frequency: 1000000000 Hz
```

Fig 6 Win32 console application for PC parallel-port control



```
38400 bps COM4 N81 - HyperTerminal
File Edit View Call Transfer Help
-----
Atmel AVR test bed for DDS/PLL hybrid synthesizer
V0.90 of Sep 30 2002 by John Miles (ke5fx@qsl.net)

FRQ n sets frequency in Hz (3- or 4-digit value treated as MHz)
-----

Command: frq 666
Setting frequency=666000000 Hz ... Locked in 2 milliseconds
Processed by AVR in 22 milliseconds

Command: frq 1000
Setting frequency=1000000000 Hz ... Locked in 8 milliseconds
Processed by AVR in 22 milliseconds

Command: frq 1296
Setting frequency=1296000000 Hz ... Locked in 9 milliseconds
Processed by AVR in 21 milliseconds

Command: _

Connected 3.24.17 TTY 38400 8-N-1 SCROLL CAPS NUM CapsLock Print echo
```

Fig 7 Communicating with the Atmel ATmega128 control program in an RS-232 terminal session

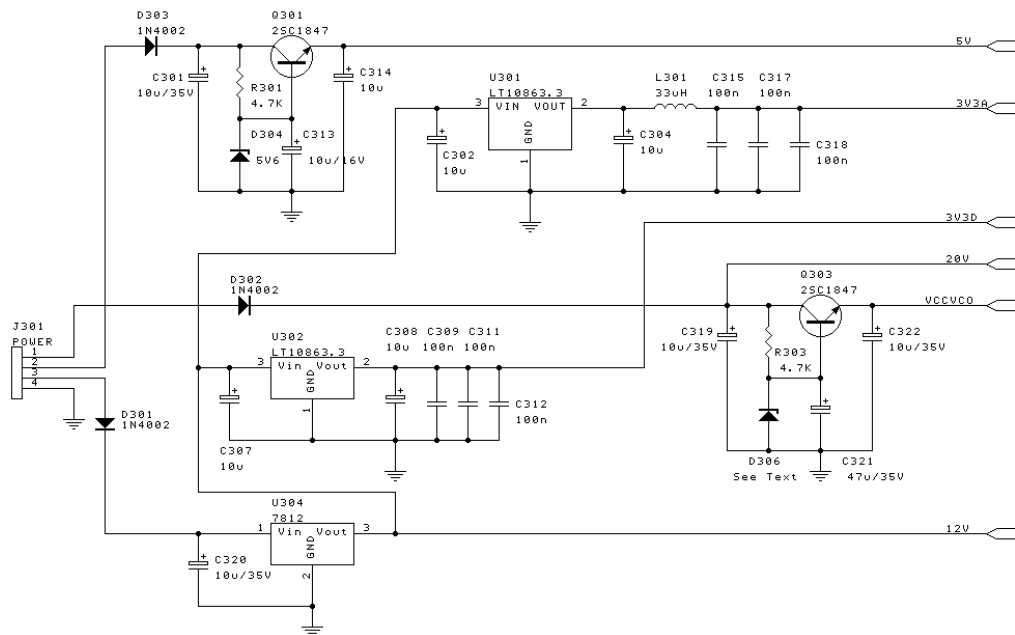


Fig 8 Power supply section

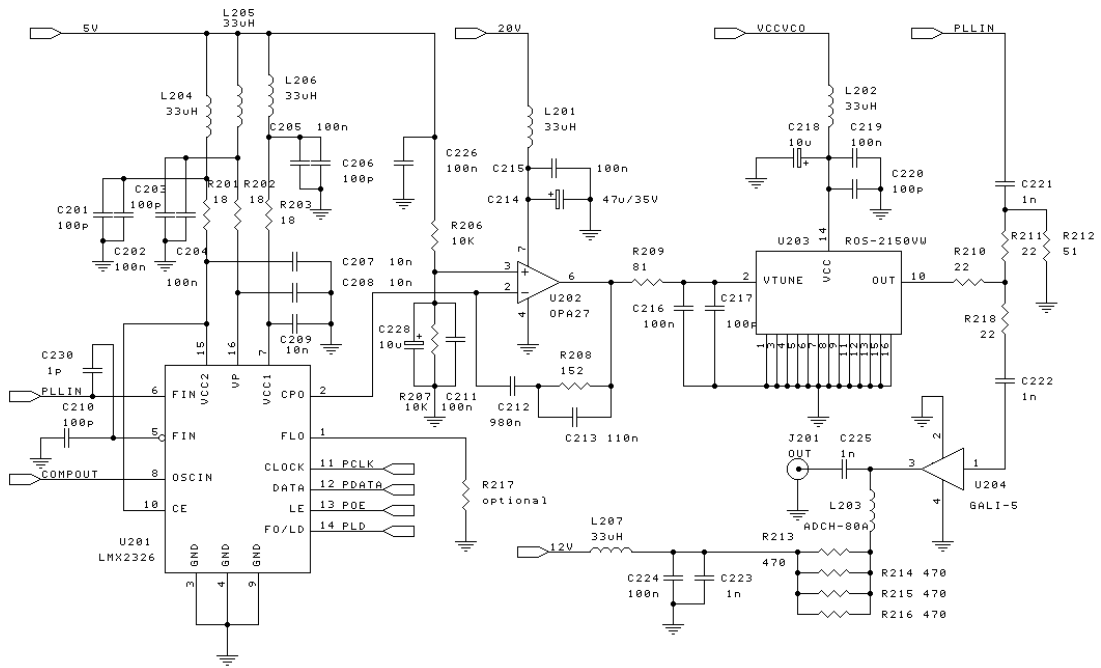


Fig 9 PLL and VCO sections

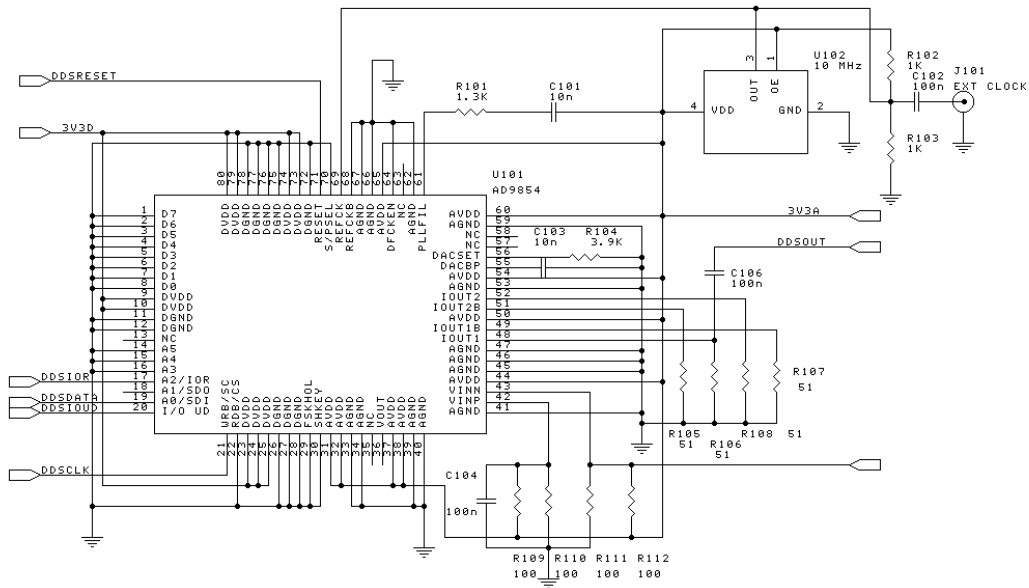


Fig 10 DDS section

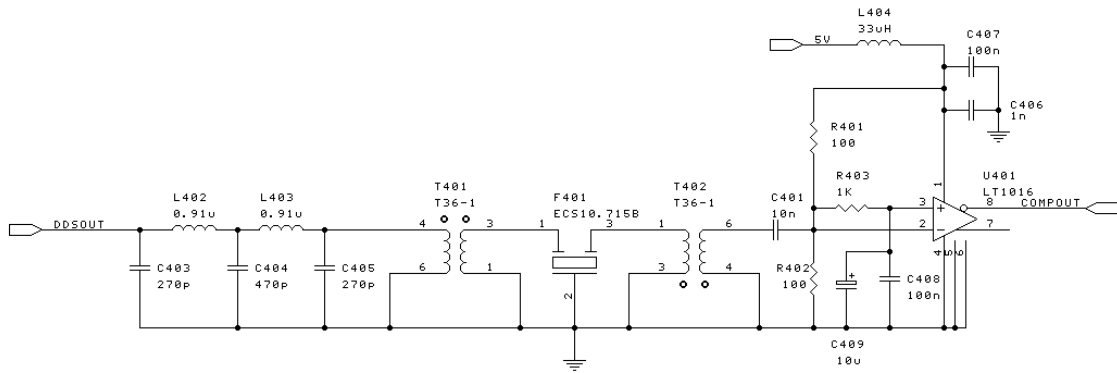


Fig 11 Filter and comparator section

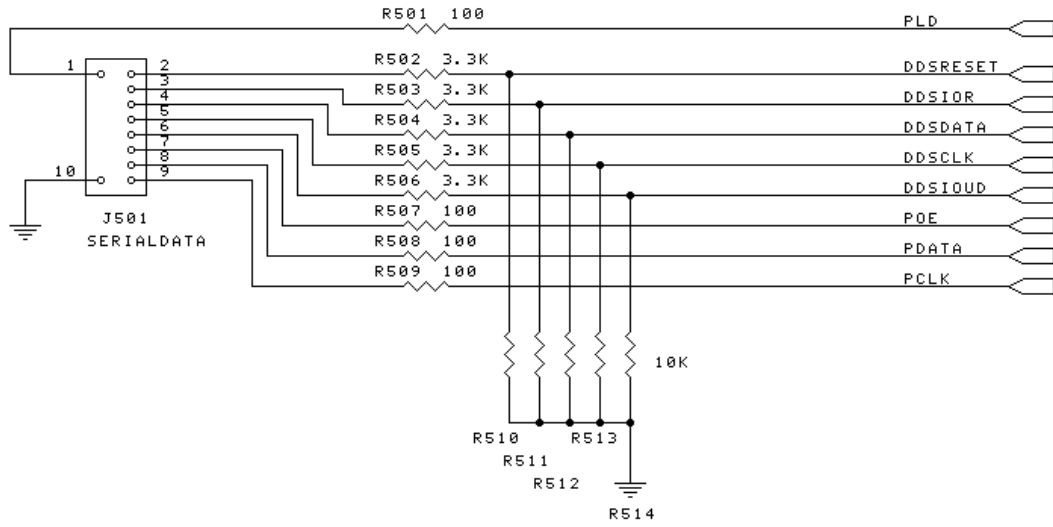


Fig 12 Data interface section

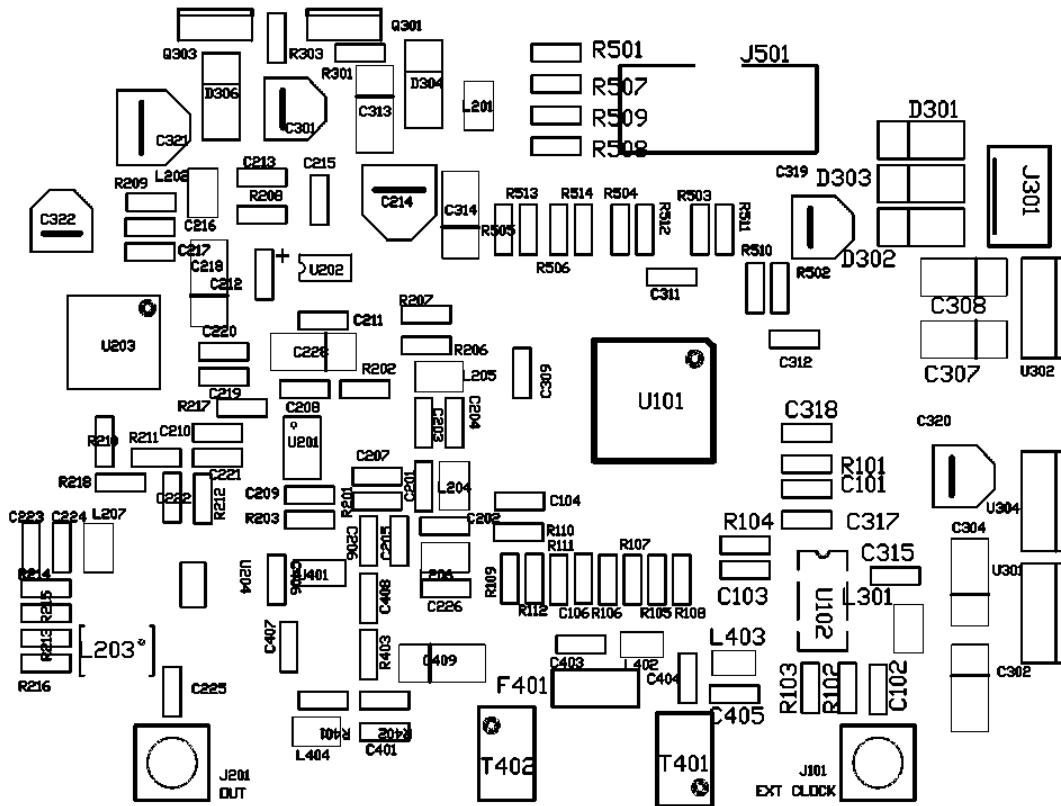


Fig 13 PCB component layout

IDC Connector	LPT Port	AVR Port 'A' Signal
1	15	PA0
2	9	PA1
3	-	-
4	6	PA2
5	7	PA3
6	8	PA4
7	4	PA5
8	2	PA6
9	3	PA7
10	18-25	GND

Table 1 PC parallel port and Atmel AVR connections to J501