Bill Kaune, W7IEQ

160 Cedarview Dr, Port Townsend, WA 98368; **w7ieq@arrl.net**

November/December 2013 *QEX*

Using GPS to Fine-Tune a Rubidium Frequency Standard

The author describes the control circuitry he developed to adjust the frequency of a Rb oscillator to lock the output signal to a GPS clock signal.

I have long been interested in instruments used to make measurements. Thus, when I saw an article in the Nov/Dec 2007 issue of *QEX* by John Raydo, KØIZ, entitled "A Low-Cost Atomic Frequency Standard," I immediately decided to build one. This device is based on a surplus LPRO-101 10 MHz rubidium oscillator module manufactured by a company named Datum. I purchased one on eBay for about $75 including shipping from Hong Kong. I purchased a 24 V, 3 A power supply from a surplus house (Marlin P. Jones, Inc.), and constructed a box out of aluminum sheet metal and a ¼ inch base plate that also served as a heat sink. I mounted these items in the box, rigged up a simple circuit that used the "lock" output from the LPRO-101 to turn on a front panel LED when the oscillator locked to the rubidium (Rb) vapor standard, and included a small cooling fan (RadioShack 273-240). I turned it on and it achieved lock after about three minutes.

I subsequently built a simple circuit that started with the 10 MHz output of the Rb frequency standard, and divided repeatedly by 2, 5, and multiples of 10. Using this circuit, I can generate Rb-stabilized frequencies from 10 MHz to 0.2 Hz in the above-mentioned steps. Over time, however, I became progressively more bothered by the fact that I had no way to independently determine the accuracy of the frequency standard, or even that it was working properly.

Then, one day I read that the global positioning satellites (GPS) use cesium clocks that are extremely accurate, and that you can purchase GPS receivers that generate time reference pulses essentially as accurate as the GPS clocks. I fired up my web browser, went to eBay, and started looking for GPS receivers. I soon found a receiver circuit board that was advertised as a Trimble Resolution-T Timing Receiver. I purchased it for $28 including shipping from Hong Kong. I also purchased a Trimble GPS antenna for $11, including shipping.

GPS Receiver

The next step was to figure out what I needed to do to get a functioning receiver at home. The first thing I discovered was that the receiver board was not actually the board that was advertised on eBay but was, instead, a Trimble GPS SMT integrated circuit installed on a "carrier" board. After reading the manual for this device, I decided it was probably superior to the Resolution-T receiver that I thought I had purchased, because of its higher-speed internal clock.

The Trimble receiver is designed to work with a computer via a "serial" interface between the two. USB ports have largely replaced the serial (COM) ports included in computers for so long. I thought it was about time for me to learn about USB ports so I purchased a book titled *USB Complete: The Developers Guide* by Jan Axelson. It did not take much reading to learn that USB ports are complicated, and I decided there had to be an easier way. I next purchased a book on serial interfaces entitled *Serial Port Complete: COM Ports, USB Virtual Ports, and Ports For Embedded Systems*, again by Jan Axelson. The key thing I learned from this book was that you could have a "USB virtual port." To do this, you buy a cable that has some embedded electronics that converts from USB at the computer end to serial at the other end and you load a software driver into the computer that has the effect of making the computer's USB port look like a serial port to any computer application software. I purchased one of these cables (Future Technologies Devices International Ltd part # TTL-232R-5V-AJ) from Digi-Key (part # 768-1068-ND) and downloaded the drivers from the FTDI website.

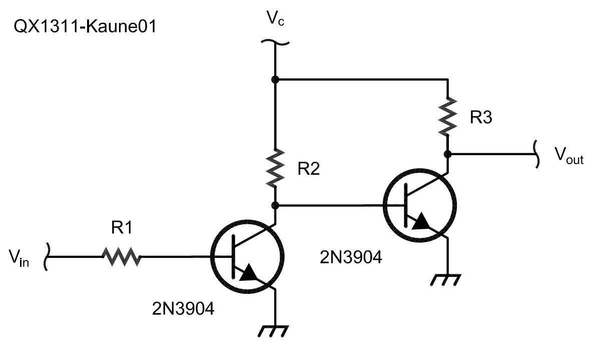


Figure 1 — This is a voltage level shifting circuit used in the GPS Interface. To shift from 3.3 V to 5 V, Vc = 5.0 V, V_{in} = 3.3 V, V_{out} = 5.0 V R1 = 100 kΩ and R3 = 220 Ω. To shift from 5 V to 3.3 V, Vc = 3.3 V, Vin = 5.0 V, Vout = 3.3 V, R1 = 150 kΩ, R2 = 7kΩ and R3 = 220 Ω.

The Trimble receiver operates with a 3.3 V power supply, whereas the low noise amplifier in the GPS antenna and the FTDI cable operate at 5.0 V. Consequently, you need two power supply voltages and two voltage level-shifting circuits, one to shift the 3.3 V serial output pulses from the GPS receiver to the 5.0 V required by the FTDI cable and the other to shift the 5.0 V serial pulses coming from the FTDI cable to 3.3 V. (I could have bought a 3.3 V cable from FTDI, but I had additional uses in mind that would require a 5 V cable.) The simple circuit I built to accomplish both of these level shifts is shown in Figure 1. The resistors R1, R2, and R3 have different values, listed in the caption, depending on the direction of voltage shift. The circuit in Figure 1 is not very fast (rise time about 0.300 μs), but it is simple and fast enough for this application. (The serial port runs at 9600 baud, which means a minimum pulse width of 104 μs.)

I bundled these two circuits, two voltage regulators and the GPS receiver board into a small metal box and named it my GPS Receiver and Interface. I downloaded from the Trimble website software named *Trimble GPS Studio*, Version 1.08.0. I connected everything together, applied power to the GPS interface, started the *GPS Studio* on my computer, and soon observed that the GPS unit seemed to be working fine. Wonders never cease!

The Trimble GPS receiver outputs a pulse every second that is synchronized nearly exactly with the GPS cesium clock. I say *nearly* exactly because in this receiver there is a little time jitter in the timing pulse, amounting to ±15.6 ns. This jitter is called "pps quantitization error" in the user manual for this device. It averages to 0 over time, and arises because the output clock pulse is synchronized with the receiver's own internal 64 MHz clock.

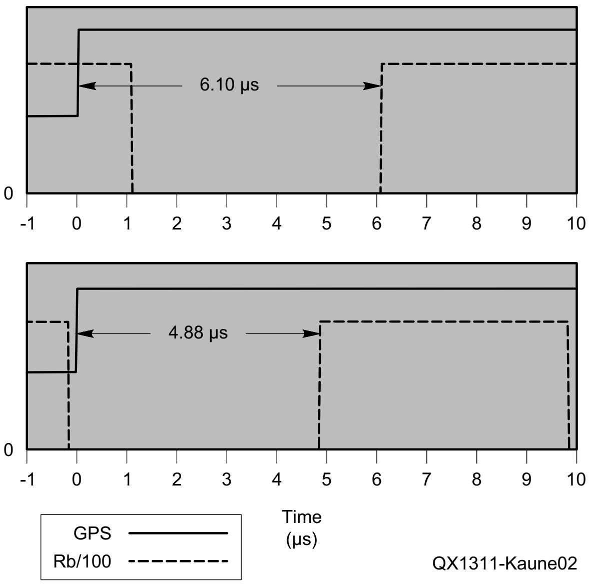


Figure 2 — Here is an oscilloscope plot showing movement in time of the Rubidium pulse relative to the GPS timing pulse during one hour.

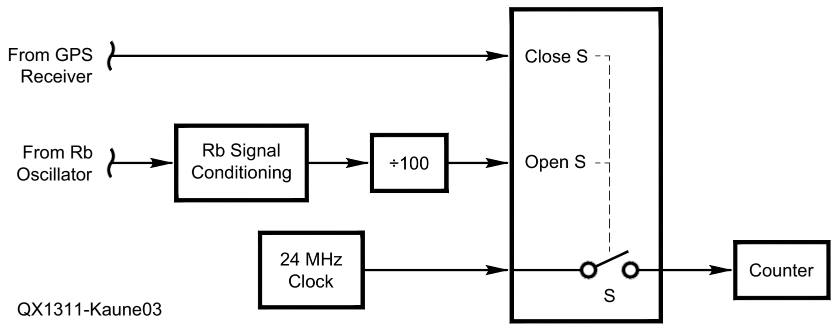


Figure 3 — This is the block diagram of the phase detector.

Measuring Frequency Error of the Rb Oscillator

I next connected the timing-pulse output of the GPS receiver to Channel 1 of my Rigol DS1052E digital oscilloscope, the output of my Rb oscillator, divided in frequency by 100, to Channel 2, and set the scope to trigger on Channel 1. The resulting display is shown in the top of Figure 2. The scope display was rewritten every second as a new GPS timing pulse arrived, but the display did not materially change. In other words, the GPS timing and frequency-divided Rb pulses were synchronized in time. This is what you would expect if the divided Rubidium frequency of 100 kHz were an exact integral multiple of the 1 Hz GPS frequency. If the Rb frequency were slightly in error — that is not exactly 10 MHz — however, one would expect that the Rb pulse shown in the oscilloscope trace would move horizontally relative to the GPS pulse over time. In fact, as I watched the traces, I observed that the Rb pulse did move slightly. The lower part of Figure 2 reproduces the oscilloscope trace one hour later than the top trace. The Rb pulse has moved to the left by an amount of 1.22 μs.

By noting the amount of movement of the Rb pulse relative to the GPS timing pulse, the error in the Rb frequency can be calculated. The change in phase, Δθ, between the GPS and Rb pulse that occurs during an elapsed time *t* is given by Equation 1.

$$\Delta\theta = -2\pi\delta f_R' t, \qquad \text{[Eq 1]}$$

where Δθ is measured in radians (2π radians = 360° phase shift) and $\delta f_R'$ is the error in the Rb frequency divided by 100, that is, $\delta f_R' = (f_R - 10\text{ MHz})/100 = \delta f_R/100$. An alternative way to measure the change in phase would be to note the time between the occurrence of a GPS pulse and the first subsequent Rb pulse. In this method, a change of phase of one cycle (2π radians) would be a change in this time difference equal to the period of one cycle, that is, $1/f_R'$ = 10 μs, where f_R' = 100 kHz is the nominal frequency of the Rb oscillator divided by 100. Equation 2 compares these two ways of measuring phase.

$$\frac{\Delta T}{1/f'_R} = \frac{\Delta\theta}{2\pi} \qquad \text{[Eq 2]}$$

where ΔT is the change in time between the GPS and frequency-divided Rb pulse that accumulates during an elapsed time t. Combining Equations 1 and 2, solving for the error in the Rb oscillator frequency, and using the fact that $\delta f'_R / f'_R = \delta f_R / f_R$, we find Equation 3.

$$\delta f_R = -f_R \frac{\Delta T}{t} \qquad \text{[Eq 3]}$$

From Figure 2, $\Delta T = -1.22 \times 10^{-6}$ s. Also, t = 60 minutes = 3600 s and f_R = 10 MHz, so δf_R = 0.0034 Hz = 3.4 mHz. In this way, I was able to determine the accuracy of my Rb oscillator.

In order for this analysis to be valid, we need to know that the Rb oscillator frequency, divided by 100, is within ± 0.5 Hz of 100 kHz; for example, a frequency of 101 kHz would also have produced a stable display. This frequency range corresponds to a range of ± 50 Hz for the Rb oscillator. Fortunately, my HP5316 frequency counter indicated the frequency of the Rb oscillator was within 1 Hz of 10 MHz.

Using GPS to Fine-Tune the Frequency of the Rb Oscillator

The LPRO-101 Rb oscillator includes a screwdriver adjustment that can be used to shift its frequency slightly. Using this capability, the frequency of the Rb oscillator could be adjusted to minimize its error. This would be relatively simple to do and would probably be adequate for nearly all uses of the Rb frequency standard. I, however, decided to go further and develop a controller that would continually monitor

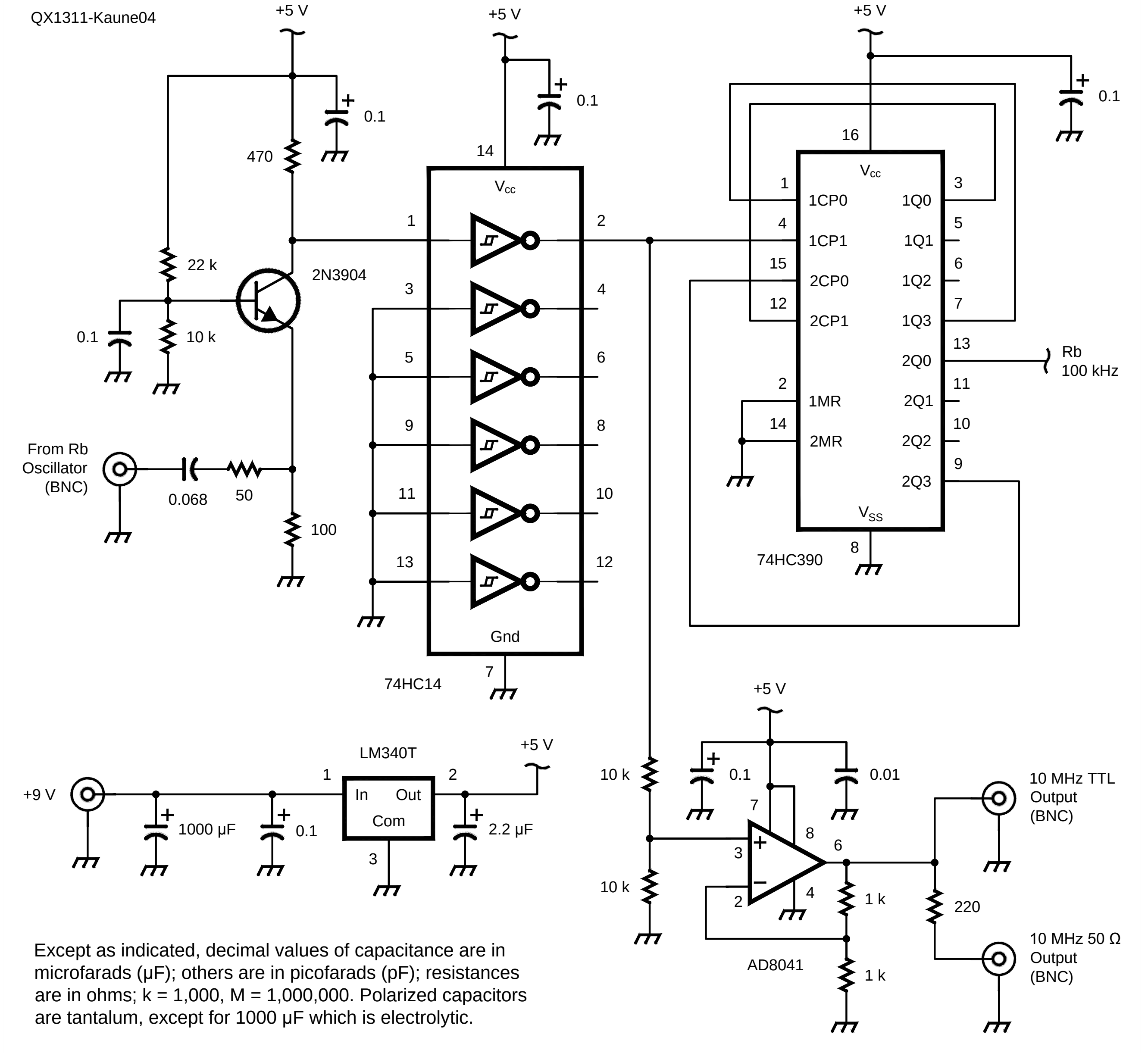


Figure 4 —The schematic diagram of the Rubidium oscillator signal conditioning, frequency dividing, and power supply circuitry.

the time difference between the GPS pulses and the subsequent Rb pulses and adjust the frequency of the Rb source to hold this time difference as constant as possible. I decided to do this because it sounded interesting and I thought I would learn a lot.

I started looking around on the Internet and soon came across a July, 1998 *QST* article titled "A GPS-Based Frequency Standard" by Brooks Shera, W5OJM. This paper describes a system that uses GPS time information to control the frequency of an oscillator. I read through it and then started designing my system. The first challenge was to design a circuit that could measure the time difference between the occurrence of a GPS pulse and the first following Rb pulse. I followed closely Brooks' strategy. Figure 3 is a conceptual diagram showing the phase detector. The occurrence, once per second, of a pulse from the GPS receiver turns on switch S. Pulses from the 24 MHz clock start to be counted. Counting continues until the next Rb pulse, divided in frequency by 100, arrives, which opens switch S and terminates the count. Once the count is complete, it will be read by a microprocessor (not shown in Figure 3) and the counter reset to zero in preparation for the next GPS pulse.

Figures 4 and 5 show the schematic diagrams of the circuits that I developed for the GPS controller. The Rb signal conditioning and divide-by-100 electronics are shown in Figure 4. The signal from the Rb oscillator is a sine wave with a peak magnitude of about 0.8 V when terminated in 50 Ω. This signal is amplified by the 2N3904 stage in Figure 4 and is routed to a Schmitt trigger inverter (74HC14) that converts it into a 5 V peak square wave. This signal is sent to both a dual decade counter (74HC390) and an AD8041 amplifier. The counter is configured to divide by 100. The output of the amplifier provides both a square wave output with peak amplitude of 5.0 V, suitable to drive most types of logic circuitry, and a 0.9 V peak output suitable for 50 Ω loads; these outputs are available on the front panel of the controller.

Figure 4 also shows the voltage regulator that provides +5.0 V for all circuitry. I use a 9 V, 300 mA wall-wart to power the voltage regulator. Because most wall warts have poor filtering, I placed a 1,000 µF electrolytic capacitor across the input.

The switch S and associated open and close circuitry shown in Figure 3 were implemented with the dual J-K flip flop (74HC109) and two-input NOR gate (74HC02) shown in Figure 5. Both flip flops are initially off. The positive-logic output, 1Q (pin 6), from the first flip flop is applied to the reset input, 2R– (pin 15), of the second flip flop. Since the reset input is *negative* logic, the flip flop is held in reset and will not respond to any inputs on its clock (2CP, pin 12). The negative-logic output of flip flop 2 (2Q–, pin 9) is applied to the reset pin of flip flop 1 (1R–, pin 1). Since this level is high when flip flop 2 is cleared, flip flop 1 is armed and will respond to pulses arriving at its input.

The J-K inputs of both flip flops are connected so that the *rising edge* of any input causes the flip flop to set. Now suppose a GPS pulse arrives at the clock input (1CP, pin 4) to the first flip flop. This raises the output 1Q high, and removes the reset from flip flop 2. Flip flop 2 then responds to the next Rb pulse, divided in frequency by 100, that arrives at its clock input. When this pulse arrives, its negative-logic output, 1Q–, goes low, resetting flip flop 1, which then resets flip flop 2.

In this way, a strobe pulse, 1Q, is generated that is high during the time interval between the leading edge of a GPS pulse and the leading edge of the next Rb pulse. The inverse of this pulse, 1Q–, is applied to one input of a dual-input NOR gate. The output of this gate will be high only if both inputs are low. The 24 MHz clock (ECS Inc. ECS-2100A-240, Digi-Key part # X221-ND) is applied to the other input. Thus, during the strobe pulse, the output of the NOR is high whenever the clock pulse is low. The 74HC393 eight-bit binary counter counts the clock pulses that make it through the NOR gate. An eight-bit counter is sufficient because the longest time the NOR gate can be open is 10 µs, which gives a maximum count of 240.

The peak voltage of the timing pulse from the Trimble receiver is 3.3 V. I initially planned to use a circuit to shift this level up to 5 V, but I discovered that 3.3 V was sufficient to reliably trigger the J-K flip flop. Consequently, in Figure 5, the input timing pulse is sent directly to the J-K flip flop.

The strobe pulse for the J-K flip flop is also sent to the interrupt input (pin 21) of a PIC 16F876A microprocessor. Whenever the processor receives this pulse, it stops whatever it is doing, waits until the count is completed, reads the 8-bit value, and sets the counter back to 0. The processor uses this and previous counts to determine a correction to the Rb oscillator frequency and outputs this correction factor to a 12-bit digital-to-analog converter (DAC7611 in Figure 5). The resulting analog value, whose range is from 0 to 4.095 V, is amplified by the OPA342 stage to cover 0 to 5.0 V, the range needed for the Rb oscillator frequency control, and is sent to the Rb oscillator to fine tune its frequency. I selected the OPA342 operational amplifier because it was advertised as a "rail-to-rail" amplifier, which means it is capable of output voltages ranging all the way from its negative supply voltage (in our case 0 V) to its positive supply voltage (5 V).

Much of the frequency controller logic is implemented in the software for the PIC processor. Figure 6 is a block diagram of the overall process. (This diagram is, essentially, the diagram of a phase-locked-loop.) The phase detector, implemented in hardware (Figure 5), determines the phase count every second in response to an incoming GPS pulse. The PIC processor reads these values and averages 120 of them during a period of 2 minutes; the average value is denoted C in Figure 6. C is compared to a target value, C_t, and a digital "error" voltage, V_1, is generated according to Equation 4.

$$V_1 = G\left(C - C_t\right) + V_{off} \qquad \text{[Eq 4]}$$

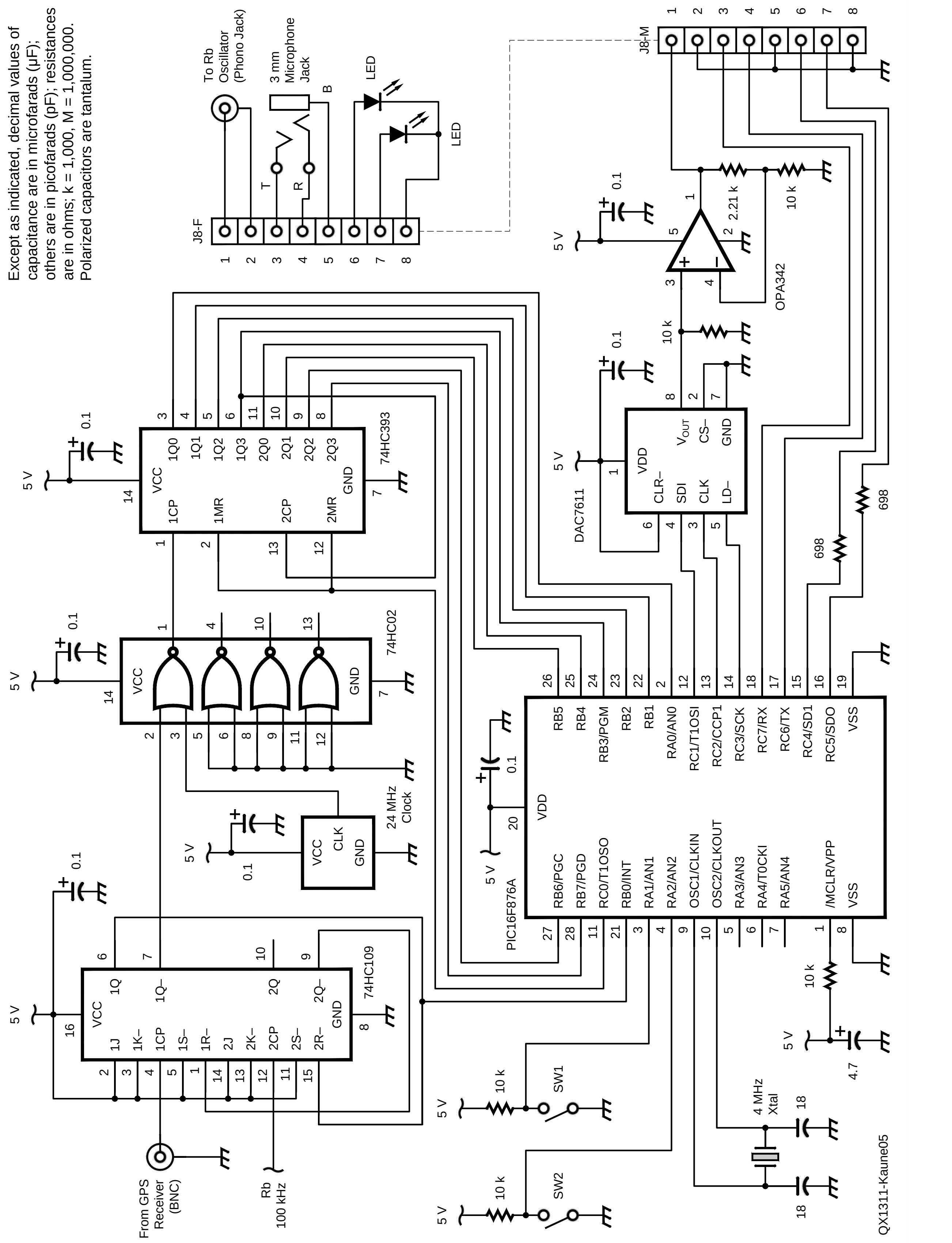
where G is the "gain" of the phase detector and V_{off} is an offset voltage. The target count, C_t, is set equal to the current count at the time that a front panel switch, discussed later, is moved from the "Open Loop" to the "Close Loop" position. The offset voltage was initially set to 2.5 V and was later changed to 2.155 V. The output digital voltage, V_1, from the phase detector is passed through a low-pass single-pole digital filter in the processor software to filter out noise and to shape the response. The digital output is converted to an analog voltage in hardware (Figure 5) and sent to the Rb oscillator.

When I started this project, I knew little about digital filters. I got a book on the subject and quickly learned that designing digital filters can be a complicated business involving Laplace and z transforms. What I needed for this project, however, was a simple low-pass filter and while fiddling around with this, I found a simple design procedure. The input to this filter is a series of numbers, x_0, x_1, x_2, and so on. This series can be denoted x_i, $i = 0, 1, 2, ...$ The filter output will also be a series of numbers, y_i, $i = 0, 1, 2, ...$ I experimented with two different filter equations, given as Equation 5A and Equation 5B.

$$y_i = Ax_i + By_{i-1} \qquad \text{[Eq 5A]}$$
$$y_i = A\left(x_i + x_{i-1}\right) + By_{i-1} \qquad \text{[Eq 5B]}$$

where A and B are constants. Using digital filter terminology, the first filter will be called the "one-tap" filter because it uses only one value of x, and the second filter will be called the "two-tap" filter. The question I addressed was how to choose A and B for each filter to obtain a filter with the desired time constant (and bandwidth).

The classic analog single-pole low-pass filter is a series resistor-capacitor. The input voltage is applied across the two and the output is taken across the capacitor. If a



Except as indicated, decimal values of capacitance are in microfarads (µF); others are in picofarads (pF); resistances are in ohms; k = 1,000, M = 1,000,000. Polarized capacitors are tantalum.
To Rb Oscillator (Phono Jack)
3 mm Microphone Jack
LED
LED
J8-F
J8-M
OPA342
DAC7611
74HC393
74HC02
74HC109
PIC16F876A
24 MHz Clock
4 MHz Xtal
SW1
SW2
From GPS Receiver (BNC)
Rb 100 kHz
5 V
0.1
10 k
2.21 k
698
4.7
18
QX1311-Kaune05

Figure 5 — Here is the frequency controller schematic diagram.

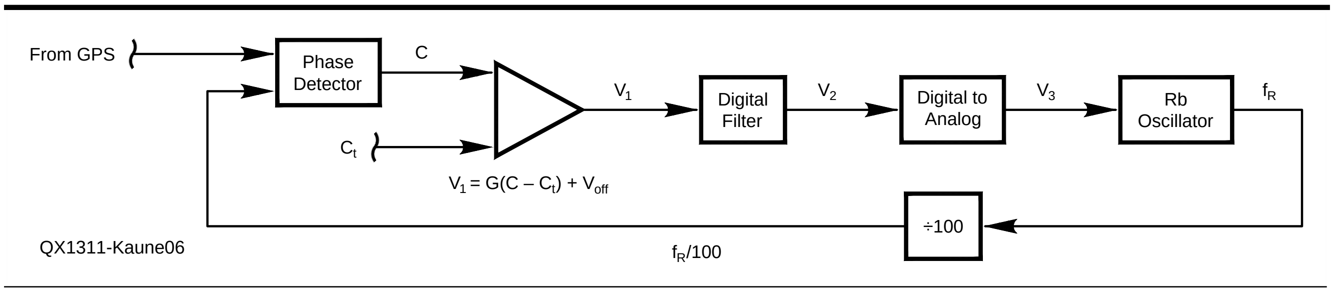


Figure 6 — This block diagram shows the frequency control logic.

steady dc voltage of 1 is applied, the output will equal the input. Our digital filter should act in the same way. Thus, if $y_{i-1} = 1$ and $x_i = 1$ for all i, then y_i must also equal 1. Placing these values in Equations 5A and 5B, we get Equations 6A for the one-tap filter and 6B for the two-tap filter.

$$1 = A + B \qquad \text{[Eq 6A]}$$
$$1 = 2A + B \qquad \text{[Eq 6B]}$$

Now suppose that a steady voltage of 1 is applied until time 0, at which point the voltage is instantly reduced to 0. In the case of the series resistor-capacitor filter, the voltage across the capacitor will decay exponentially with a time constant τ. That is expressed as Equation 7.

$$y = e^{-t/\tau} \qquad \text{[Eq 7]}$$

where t is time > 0 and τ is the filter time constant.

For the digital filter, $x_i = 0$ for $i \leq 0$ and $y_0 = 1$. Using Equation 5A or 5B repeatedly:

$$\begin{aligned} y_1 &= By_0 = B \\ y_2 &= By_1 = B(B) = B^2 \\ y_3 &= By_2 = B(B^2) = B^3 \\ &\vdots \quad \vdots \quad \vdots \end{aligned}$$

Therefore, we see that

$$y_N = B^N \qquad \text{[Eq 8]}$$

where N is related to time. If the time interval between samples is T_S, then $N = t / T_S$. Thus, Equation 8 becomes Equation 9.

$$y = B^{t/T_S} \qquad \text{[Eq 9]}$$

In order for the responses of the analog [Equation 7 and digital Equation 9] filters to be the same, we need the condition established by Equation 10.

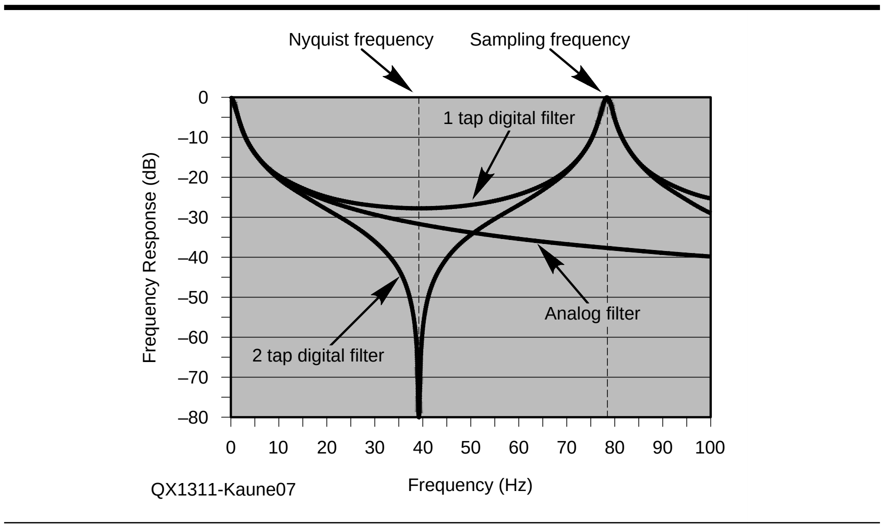


Figure 7 — This plot shows the frequency responses of the digital filters and equivalent analog filter.

$$B^{t/T_S} = e^{-t/\tau} \qquad \text{[Eq 10]}$$

This equation may be solved for B and Equation 6 used to calculate the value for A.

$B = e^{-T_S/\tau}$ and $A = 1 - B$ for the one-tap filter [Eq 11A]

$B = e^{-T_S/\tau}$ and $A = (1 - B) / 2$ for the two-tap filter [Eq 11B]

In this way, we have related the parameters of the digital filter to those of its analog counterpart.

I next tested these two filters using computer-generated sine-wave inputs. Figure 7 shows the frequency responses of both digital filters and a corresponding analog filter for a time constant chosen to give a bandwidth of 1 Hz and a sampling frequency of 78.57 Hz. (The sampling frequency was chosen to yield the same product of the time constant and sampling frequency as the filter used in the final GPS frequency controller.) The frequency response of the analog filter decreases steadily with frequency. Both digital filters have essentially the same frequency response as their analog counterpart up to about 20 Hz. Above this frequency, the response of the one-tap filter starts to flatten out while the response of the two-tap filter steepens. At ½ of the sampling frequency, called the Nyquist frequency, the response of the one-tap filter is flat while the two-tap filter becomes almost zero. Things get more complicated as the input frequency, f, is increased above the Nyquist frequency. The temporal pattern of the samples of this input is exactly the same as the pattern obtained when sampling an input frequency of $2f_N - f$ (f_N = Nyquist frequency). Thus, the response of the filter is the same for input signals with frequencies of f and $2f_N - f$, with the result that the magnitude of the filtered signal rises, reaching 1 (0 dB) at twice the Nyquist frequency. Figure 7 shows that the response of the filter is symmetrical around the Nyquist frequency.

I was initially quite surprised to find that the attenuation of the two-tap filter was so great near the Nyquist frequency until I realized that at the Nyquist frequency, $x_i + x_{i-1} = 0$ for all i. I selected the two-tap digital filter for use in the frequency controller software.

The output from the digital filter is converted to an analog voltage in hardware (Figure 5), amplified slightly, and used to fine-tune the Rb oscillator. I measured the oscillator frequency error for different control voltages using the method described earlier and summarized by Equation 3. Figure 8 shows the results. The measured points are shown as plus signs, and the straight line is the best linear fit to these data. This line is given by Equation 12.

$$\delta f_R = -0.0188 + (0.0090)V \qquad \text{[Eq 12]}$$

where δf_R is the frequency error of the Rb oscillator in hertz, and V is the control voltage applied to the oscillator. (Actually, the data show a slight upwards curvature and a slightly better fit is provided by the quadratic form $\delta f_R = -0.0178 + 0.00775\ V + 0.00025\ V^2$.)

I wrote assembly-language software for the PIC processor to implement Figure 6 and to also continually send phase counts, averaged over the two minute measurement period discussed earlier, and control voltages to a computer, connected to the PIC via the same serial cable used earlier to connect the GPS Receiver-Interface to a computer. The data sent to the computer enabled me to monitor the GPS frequency controller operation. I spent considerable time developing this software and experimenting with different values for the phase detector gain, G, that appears in Equation 4, and the time constant, τ, of the digital filter. In selecting these values, I obtained guidance from phase-locked-loop theory. I also had to develop a method of determining when the Rb oscillator became "locked" to the GPS timing pulse.

My final selections for G were 0.06 before lock was achieved and 0.012 after. For the filter time constant, I selected 300 s before lock was achieved and 1500 s after. I found by having two selections for each of these parameters, I could substantially reduce the time to the achievement of lock. Once lock was achieved, the increase in τ and corresponding decrease in G provided increased immunity against noise.

I found that a suitable way to determine lock status was when the variation in the phase count became sufficiently small. My ultimate algorithm takes a consecutive sequence of 11 phase count measurements, covering a period of 22 minutes, and determines their standard deviation. Lock was achieved when this standard deviation became sufficiently small.

Figure 9 is a photograph of the front panel of the controller. The left-most switch can be used to place the controller in a WAIT mode, where it does nothing. In the RUN position, the controller starts to make phase measurements and send these data to a computer connected to it. The next switch determines whether the controller will ("Close Loop") or won't ("Open Loop") adjust the control voltage to achieve phase lock. The left-most two BNC connectors provide the 10 MHz 50 Ω and TTL outputs mentioned earlier. The right-most two BNC connectors are inputs for the Rb oscillator and GPS timing pulse, respectively. The two LEDs above the BNC connectors, marked L1 and L2, show the status of the controller. If both are off, the controller is in WAIT mode. If L1 is blinking once a second, the controller

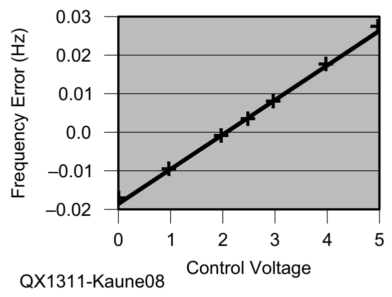


Figure 8 — This graph shows the Rubidium oscillator frequency error as a function of the control voltage. Measured data are shown as "+" signs. The straight line is a linear fit to the measured data.

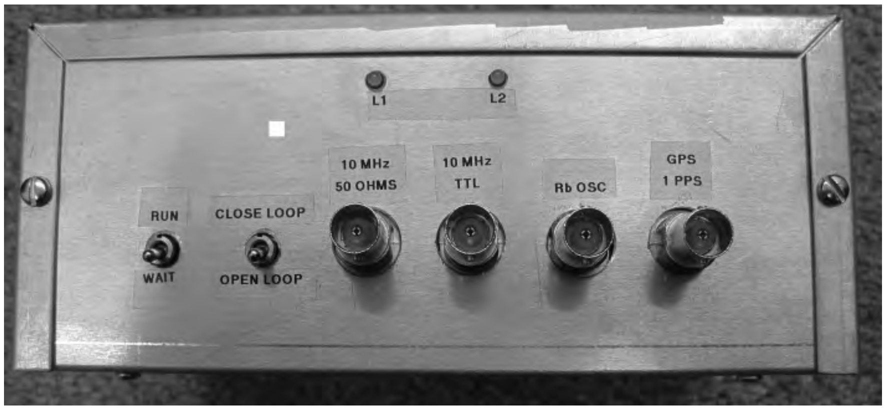
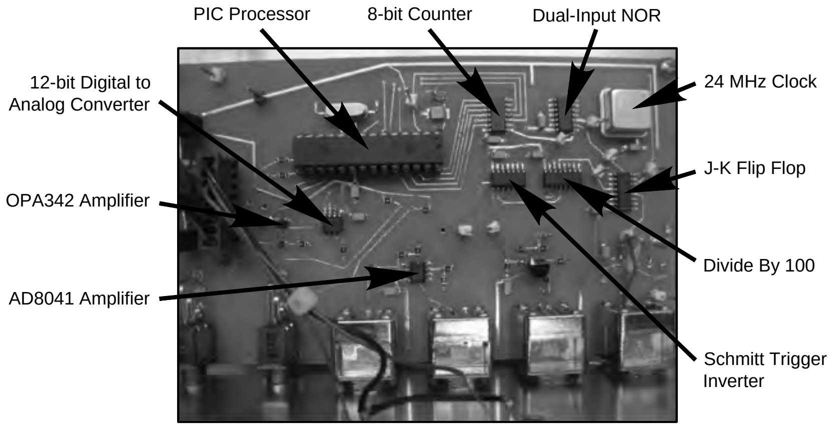


Figure 9 — Here is a photograph showing the front panel of the Rubidium-GPS frequency controller.



QX1311-Kaune10

Figure 10 — This shot shows the frequency controller circuit board with the various components labeled.

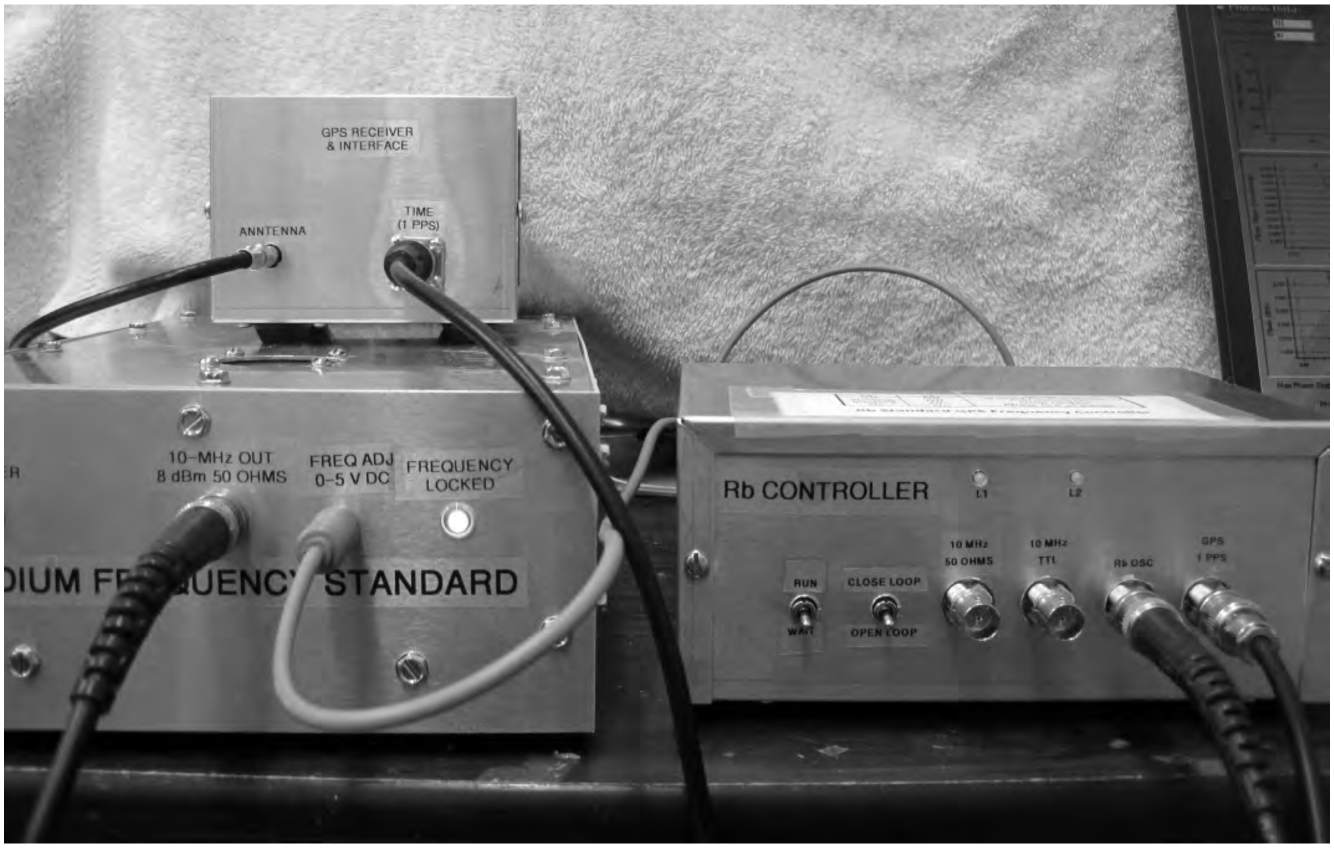


Figure 11 — Here is a view of the complete Rubidium oscillator frequency control system.

is in RUN mode. If, in addition, L2 is lit, the controller has locked the frequency of the Rb oscillator to the GPS timing pulses. Finally, if both lights are steadily on, the controller has found the phase count too close to its minimum value (0) or to its maximum value (240). In this case, the controller will adjust the control voltage to most rapidly move the phase count into an acceptable range.

Figure 10 is a photograph of the circuit board of the controller with most of the major components labeled. Most components were surface mounted; the PIC processor was not since I needed to be able to remove it for programming. (If you look closely, you can see two places on the board where I made corrections.) The two switches and the four BNC connectors were mounted directly on the computer board. The control voltage output, the 9 V dc power input, and the serial input/output jack were mounted on the back cover of the enclosure. As you can see from Figure 10, I could have made the board considerably smaller and more compact, and thus saved on fabrication costs.

Figure 11 is a photograph of the entire system. The Rb oscillator is at the lower left in the picture. Sitting on top of it is the GPS Receiver and Interface unit. The frequency controller is at the lower right. You can just see the left-hand part of the computer screen being used to monitor the system.

Testing the Complete System

For testing I connected my notebook computer to the frequency controller through a serial interface as described earlier. I wrote software for the computer using *Visual Basic* Version 6.0. This version of *Visual Basic* includes a serial port (COM) "control" that implements a software interface to the serial port. My application software receives phase count and control voltage data every 2 minutes and additional information regarding when the front-panel Open/Close Loop switch is placed in the Close Loop position and when lock is achieved. The software displays six graphs: Phase count versus time; control voltage versus time; rate of change of phase count versus time; rate of change of control voltage versus time; residual jitter in phase count versus time; and residual jitter in control voltage versus time. The software can also be used to save data for later off-line analysis.

I tested the system in various ways. Figure 12 shows the phase count and control voltage as functions of time during one test where I initially set the control voltage to 0 to induce an initial Rb oscillator frequency error of about –17 mHz. The figure shows an initial transient period, lasting a bit more than 1 hour, where the control voltage is adjusted upwards, initially overshooting its final value of about 2.15 V. The system determined that frequency lock was achieved at 1.45 hours. I could have reduced or eliminated the overshoot by adjusting the phase detector gain and/or the digital filter time constant, but doing so would not have materially decreased, and could have substantially increased, the time to final lock.

A series of tests showed that a control voltage near 2.15 V was consistently needed to bring the error in the frequency of the Rb oscillator to 0. Consequently, I altered the PIC program to always start with an initial control voltage of 2.153 V. Figure 13 shows the results of a test after this change was made. Note that frequency lock was achieved at 0.35 hours. This graph illustrates clearly the noise in the phase count signal. There are at least three sources of noise: the

1 pps quantitization error mentioned earlier that originates from the GPS receiver, the use of a 24 MHz clock to measure time, and time jitter introduced by the digital electronics used to measure phase count. Even allowing for noise, the phase count data in Figure 13 appears to be increasing slightly during the period from lock at 0.35 h to about 5 h; beyond 5 h, the phase count seems to be on average relatively steady. Statistical (regression) analysis confirms this impression and indicated that the slope of phase-count versus time was (0.122 ± 0.009) counts per hour during the period 0.35 h < t < 5 h and was consistent with 0 for $t \geq$ 5 h.

A change of phase count of 0.122 in one hour indicates that the Rb oscillator frequency was slightly in error. The maximum phase count change, corresponding to a change of phase of 2π radians, is 240. Thus, the fractional change in phase, during the one hour time period, was 0.122 / 240 = 5.08×10^{-4}. Equation 1 also expresses the change in phase, measured in radians, that would occur during a time period of length t resulting from an error, δf_R, in the frequency of the Rb oscillator: $\Delta\theta / 2\pi = -\delta f_R\, t$. Combining these two, $\delta f_R = -0.122 / (240 \times 3600 \text{ s}) = -0.14$ μHz, a very small error.

The control voltage, shown in Figure 13, exhibits a slight increase during the period 0.35 h < t < 5 h, reflecting the correction necessary to halt the slight increase in phase count discussed in the preceding paragraph. Beyond 5 h, the control voltage is relatively stable. The residual variation in the control voltage, after accounting for the long-term slopes, is about 1.2 mV. From Equation 12, we see that a change in control voltage of 1 V will produce an approximate change in the frequency of the Rb oscillator of 9 mHz. Thus, the residual random variation of the Rb oscillator frequency error is about 1.2 mV × 9 mHz / V = 11 μHz. Based on this and the earlier results, I think the long-term agreement of my Rb oscillator with the GPS standard, when using the GPS system to fine tune its frequency, is about ± 20 μHz, that is, 2 parts in 10^{12}. I do not have the equipment necessary to measure the short-term time jitter, or equivalently phase noise, of my Rb oscillator. Others have done these measurements, however, and report phase noise at 0.1, 1, 10, 100, and 1000 Hz and greater of –60, –76, –96, –135, and < –150 dBc/Hz, respectively.

My main motivations for this project were to determine that my Rb frequency standard was working properly and also as an educational undertaking. I learned a number of things, especially about the GPS system, serial interfaces, digital filters, and phase-locked-loops. Also, I now have a frequency standard that is extremely accurate. I use it to calibrate an HP frequency counter I purchased on eBay, and my transceiver. I cannot calibrate these units to accuracies any better than 0.1 Hz and 10 Hz, respectively, so the fact that my standard is accurate to ± 20 μHz instead of 3 mHz is really not important. Lately, I have been thinking about designing and building a receiver for use in the ARRL's periodic frequency measuring test that could make better use of this frequency standard.

Bill Kaune, W7IEQ, is a retired physicist (BS, 1966; PhD, 1973). He is married, has two grown daughters, four grandchildren, and a standard poodle. Bill spent most of his career collaborating with biologists and epidemiologists researching the biological effects of power-frequency electric and magnetic fields. Along with Amateur Radio, Bill spends his time hiking and backpacking. Bill was first licensed in 1956 as a Novice and then a General, became inactive while in college, and was licensed again in 1998. He upgraded to the Amateur Extra class in 2000. Bill is currently the president of the Jefferson County Amateur Radio Club, a member of the ARRL and the ARRL RF Safety Committee, and a fellow of the IEEE.

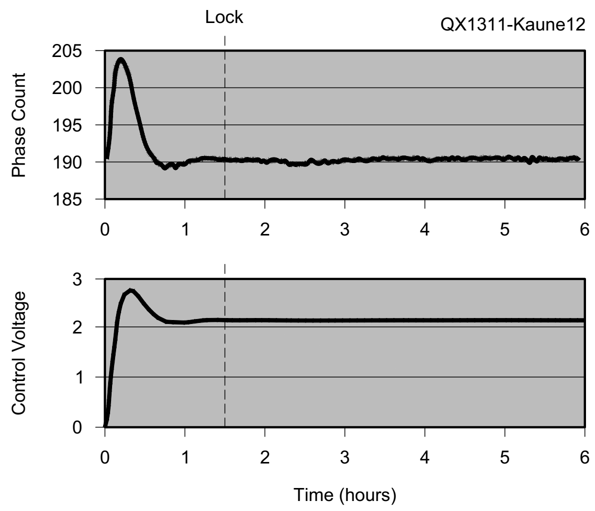


Figure 12 — Test run of the Rubidium-GPS frequency control system. The initial frequency error was –17 mHz.

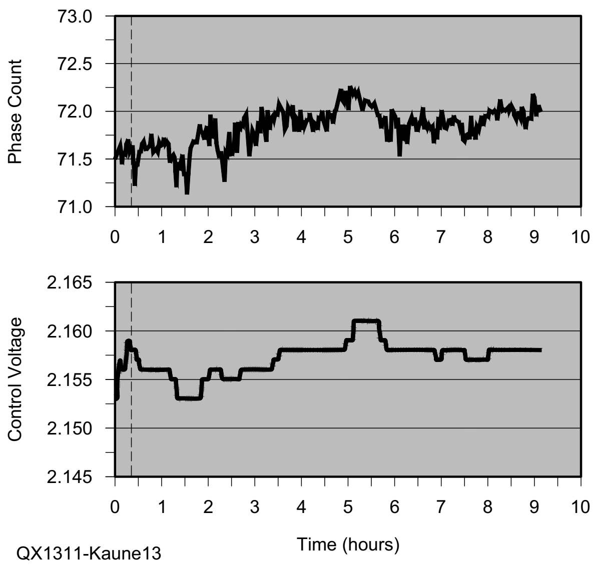


Figure 13 — Here is another test run of the Rubidium-GPS frequency control system. The initial frequency error was very small.