Gary Richardson, AA7VM

PO Box 228, Marblemount, WA 98267; **aa7vm@arrl.net**

An RF Filter Evaluation Tool

Here is a microprocessor controlled system to measure filter frequency response.

Unless you have a spectrum analyzer, determining the response of an RF filter is rather tedious — adjust the signal generator, measure the output, record the value and repeat until sufficient data has been collected. Then you must convert the values to decibels and make a plot, or enter the data in your computer and use a program to generate the plot. Not long ago I was occupied with this drill, and it seemed to me that there ought to be a way to automate the process. A little looking around turned up a lot of information on the subject of RF signal measurement.[1, 2, 3, 4] After several false starts I came up with a design based on the AD8307 that seemed promising.

A block diagram of my measurement system is shown in Figure 1. It consists of a computer having at least one serial port, a signal generator that can be controlled by commands sent to its serial port, the filter to be evaluated (DUT), a couple of attenuators and a bit of hardware to tie everything together: a detector module and a control module. The main component of the detector module is an AD8307, which generates a DC voltage proportional to the power of the input RF signal. The control module has two functions accomplished by a microprocessor: relaying command strings from the computer to the signal generator and acquiring and sending the power measurements to the computer.[5]

Figure 2 is a photograph of the two modules mounted in a small enclosure. Following the advice of Hayward and Larkin, I mounted the detector board in a smaller enclosure made of 24-gauge copper sheet (the top cover has been removed for

[1]Notes appear on page 6.

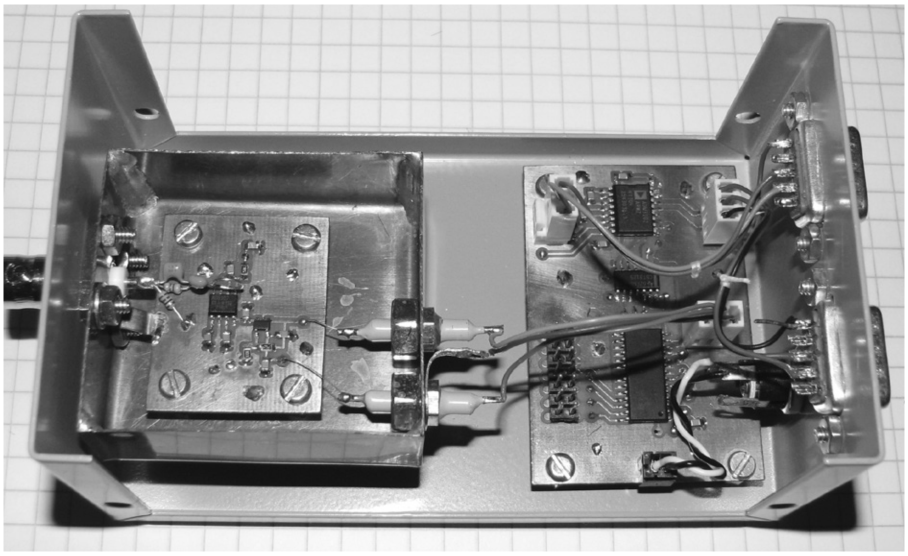
Figure 2 — The photo shows the two modules of the measurement system mounted inside a small metal enclosure. The detector board is also mounted inside a separate enclosure. For this photo, the top cover of that enclosure has been removed.

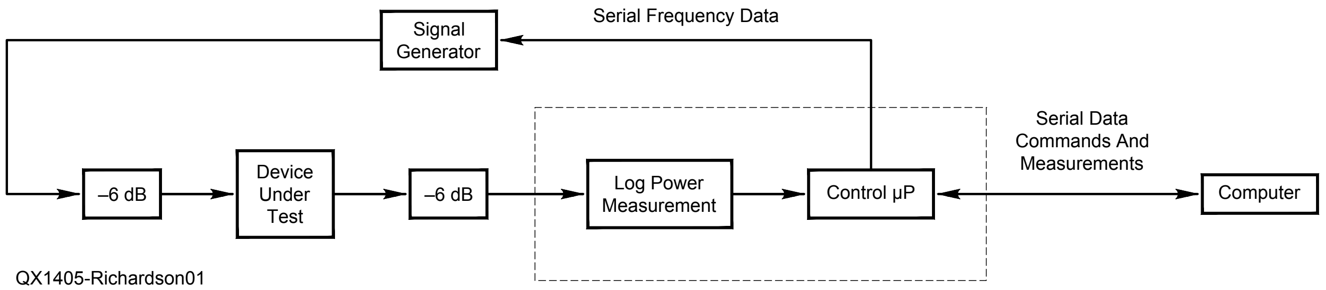

Figure 1 — This is a block diagram of the filter measurement system.

viewing in this photo). A separate enclosure for the detector board might not be necessary if the larger enclosure is RF tight.

Electronics

Figure 4 is the schematic of the detector board and is a variation of the circuit described in note 2. All of the components are mounted on the top side of a small piece of 1/32 inch double-sided circuit board. The initial version used surface-mount parts for the input network but the high-frequency response, above 100 MHz, was lower than that shown in Figure 2 of the Hayward/Larkin article and I thought small through-hole components might be an improvement. There was no significant change, however. With a 5 V V_{CC}, the output of the AD8307 ranges from about 200 mV to about 2.5 V. An op-amp with a gain of gain 1.25 increases the maximum output voltage to slightly less than the maximum ADC input of 3.3 V. I had originally planned to operate the AD8307 at 3.3 V but I found that the dynamic range was considerably reduced.

The microprocessor series I'm most familiar with is the TI MSP430, inexpensive, low power devices that have a very nice instruction set and excellent open-source development and debugging tools. I chose an MSP430F1232 for this project because it is the smallest processor in this series with an ADC (10 bits).

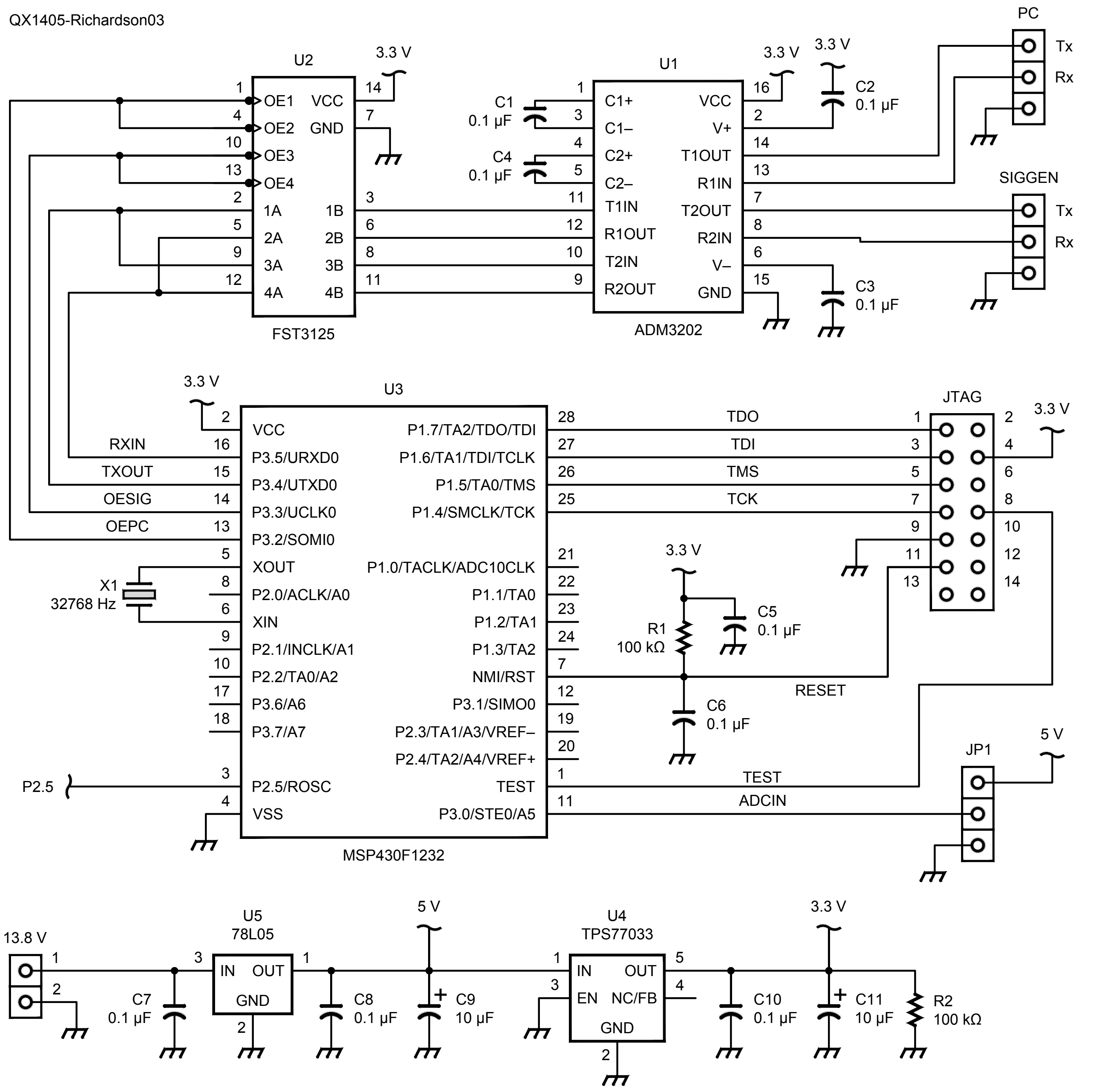

Figure 3 — Here is the schematic diagram of the microprocessor board.

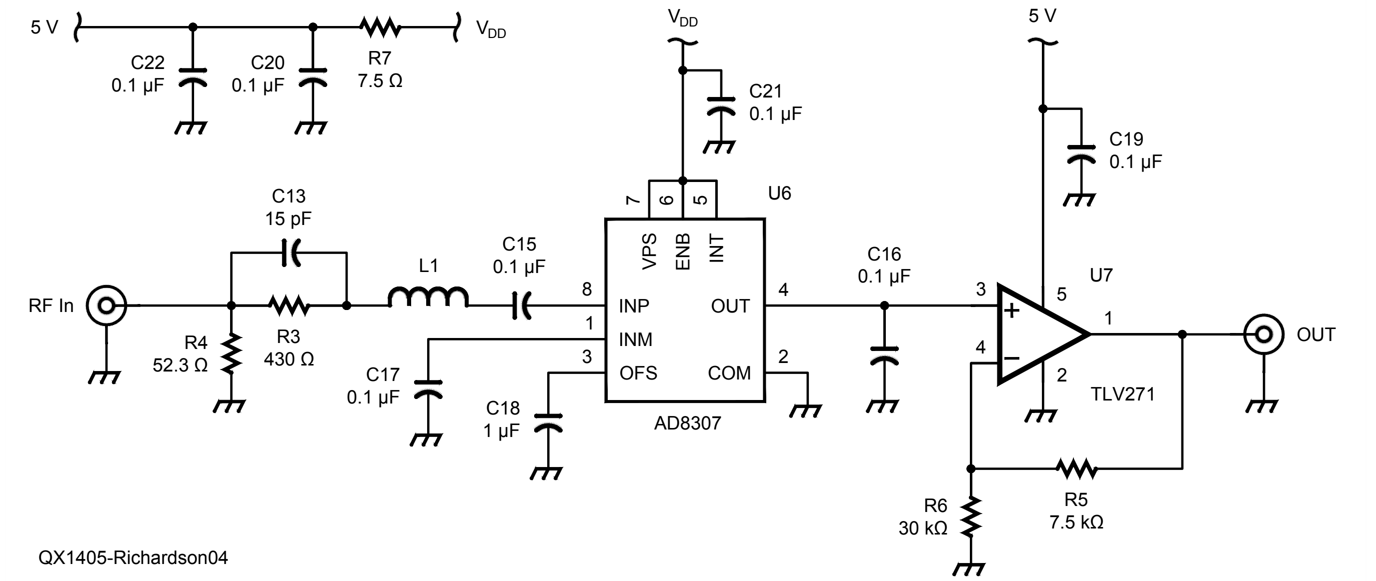

Figure 4 — This schematic diagram shows the circuit of the detector board.

Figure 3 is a schematic of the processor board. I had initially planned to include an FT245R USB/Serial IC on this board to provide a USB channel to the computer. I created a nice looking schematic but progress in routing the traces on the board quickly ground to a halt. With only two layers it was impossible (for me at least) to route the connections between the FT245R and the microprocessor. It might have been possible to connect the two devices with wires, but it would have been a mess. So I adopted a serial approach.

The ADM3202 is a dual RS-232 line driver/receiver device. The SN74CBT3125 is a quad FET switch that connects one of two serial channels to the microprocessor. This board supplies the power for both boards, hence the two voltage regulators. The total current consumption is about 23 mA.

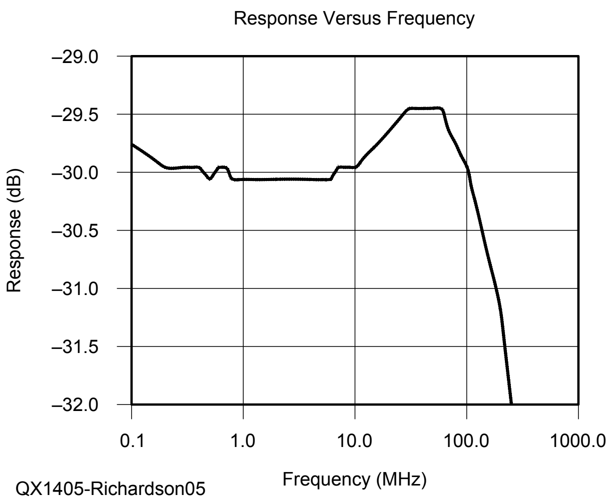

Figure 5 — This graph is the plotted frequency response of the unit. You can see that with only a few tenths of a dB variation, the response is flat from 100 kHz through 100 MHz.

Performance

Figure 5 is a plot of the frequency response of the unit for a –30 dBm input; it varies a few tenths of a dB between 100 kHz and 100 MHz, but then falls off rapidly, whereas the response of the Hayward/Larkin unit is reasonably flat out to 600 MHz. I used a BNC connector for the RF input and RG-58 coax to connect to the signal generator. I assume the bandwidth of my unit would have been greater had I used an N-type connector and high quality coax.

The top portion of Figure 6 is a plot of detector output in ADC counts versus input power for a 10 MHz input signal. A regression (least squares fit) line is drawn through the data points from –70 dBm to +15 dBm. The slope of this line is the scale factor of the measurements in terms of ADC counts per dBm. Y0 is the value on the regression line for 0 dBm. The lower part of the graph shows the difference between the data points and the corresponding values on the regression line, and provides an indication of the linearity of the AD8307.

Software

The software for the MSP430F1232 was written in *C* using the mspgcc development tools.[6] The computer software was written in *Python* and consists primarily of two modules, one dealing with the communication with the microprocessor and the signal generator and the other with various higher level functions, primarily plotting.[7] The code that generates the plot of Figure 6 also serves to calibrate the system; the gain and Y0 values are saved in a file and are used by the plotting functions. An example of the use of the plotting function is shown in the code of Figure 7. This code was used to generate the plot of Figure 5. The arguments to the plot function are obvious except perhaps for "yOffset." If this argument is set equal to the total pad attenuation (typically 12 dB) minus the signal generator power level, the peak plot values will be near zero. The function "plotFrequencyResponse" has an additional argument not shown Figure 7, *findBandwidth*, which when set to true will cause a horizontal line to be drawn at the –3 dB level and the bandwidth of the response curve to be computed and displayed on the plot. This requires that the yOffset argument be set to –12 dB (assuming 6 dB attenuators used).

The files for this project are available for download from the ARRL *QEX* files website at **www.arrl.org/qexfiles**. Look for the file **7x14_Richardson.zip**.[8] The files include the software (*Python* and microprocessor code) and the Eagle schematic and circuit board layout files.[9]

Gary Richardson, AA7VM was first licensed as KN5WHO in 1957. Interest in amateur radio waned in subsequent years due to pressure of school and work. Gary earned an MSEE degree from Michigan State University in 1967 and spent much of his career designing software for embedded microprocessors in medical systems. He was licensed as AA7VM in 1999. You can reach Gary at PO Box 228, Marblemount, WA 98267 or aa7vm@arrl.net.

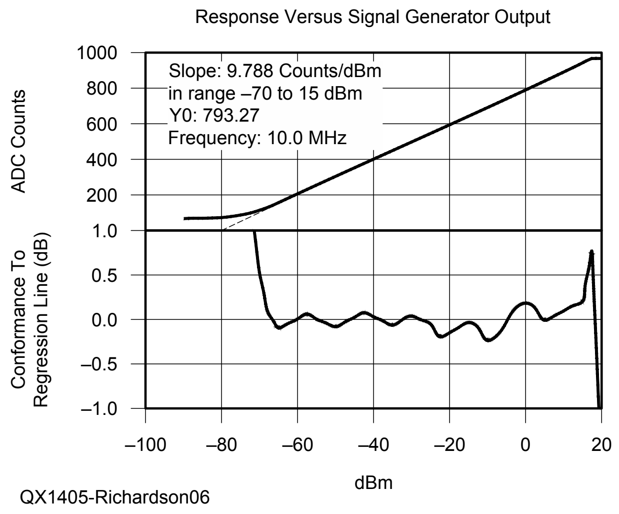

Figure 6 — The top portion of this graph shows the analog to digital converter counts versus the input power from the signal generator, which is set to 10 MHz. The lower portion of the graph shows the deviation from or the conformance of the measurements to a least squares fit of the data. You can see that the AD8307 log detector is linear from about –75 dBm through +10 or +15 dBm.

Notes

[1]Wes Hayward, W7ZOI and Bob Larkin, W7PUA, "Simple RF-Power Measurement," June 2001 *QST*, pp 38-43.

[2]Wes Hayward, W7ZOI, Rick Campbell, KK7B, and Bob Larkin, W7PUA, *Experimental Methods in RF Design*, Section 7.3. ISBN: 978-087259-923-9; ARRL Publication Order No. 9239, $49.95. ARRL publications are available from your local ARRL dealer or from the ARRL Bookstore. Telephone toll free in the US: 888-277-5289, or call 860-594-0355, fax 860-594-0303; **www.arrl.org/shop**; **pubsales@arrl.org**.

[3]Loftur Jonasson, TF3LJ/VE2LJX, "Squeeze Every Last Drop Out of the AD8307 Log Amp," May/June 2013 *QEX*, pp 29-33.

[4]For more information about the AD8307 logarithmic amplifier see the Analog Devices website: **www.analog.com/en/rfif-components/detectors/ad8307/products/product.html**

[5]If your computer has two serial ports one could be used to talk to the signal generator, eliminating one of the tasks the microprocessor would need to perform, and would simplify the hardware somewhat.

[6]For information about the mspgcc development tools see: **http://sourceforge.net/apps/mediawiki/mspgcc/index.php?title=MSPGCC_Wiki**

[7]You can find more information about *Python* at: **www.python.org/**. The plots were generated with the matplotlib package: **www.matplotlib.org/**

[8]The files for this article, including the software (*Python* and microprocessor code) and the Eagle schematic and circuit board layout files are available for download from the ARRL QEX files website at: **www.arrl.org/qexfiles**. Look for the file **7x14_Richardson.zip**.

[9]The schematic shown in Figure 4 is not exactly the same as the schematic used to generate the board because the input network components (C13, L1, R3, R4, C15) are through-hole components and are not mounted on the board, neither is the coax connector.

Figure 7 Computer Code Listing

```
from PMSA.process import PMSA_PROCESS
dirname = 'C:\Documents and Settings\Owner\My Documents\Python\PMSA'
pmsa = PMSA_PROCESS(dirname)
freqs = [0.1*k for k in range(1, 10)]
freqs += [k for k in range(1, 10)]
freqs += [10*k for k in range(1, 10)]
freqs += [100*k for k in range(1, 5)]
freqs = array(freqs)            # Frequencies have units MHz
siggen = -30                    # signal generator power level - dBm
yH = siggen + 1                 # Max Y axis limit in dB
yL = siggen - 2                 # Min Y axis limit
yOffset = 0                     # no Y-a xis offset
title = 'Response vs Frequency'
mode = 0                        # semilog plot
pmsa.plotFreqResponse(freqs, siggen, yOffset, yL, yH, title, mode)
```