Design of an HF QSD SDR for the Arduino and Raspberry Pi Platforms

Edward Cholakian[i] KB1OIE

Abstract

This paper describes the implementation of a software defined radio for HF reception to be used on two popular single board computer form factors, the Arduino, and the Raspberry Pi. The goal of the design is to produce good quality receivers that are inexpensive and open sourced for further experimentation and software development. The design includes an RF front end and bandpass filters, a quadrature sampling detector, analog to digital converters with drivers, digital signal processing (DSP), display interfaces, and digital audio stages for a complete receiver without the use of an external computer. Alternately the radios can run as remote controlled local or internet connected server of the received digitized data. A prototype for an Arduino using a 32 bit Microchip MIPS based processor with fixed point DSP has been built, and two new multilayer boards are currently being tested. The required digital signal processing algorithms have also been written in C using only double precision floating point hardware for ARM and x86 implementations.

Keywords

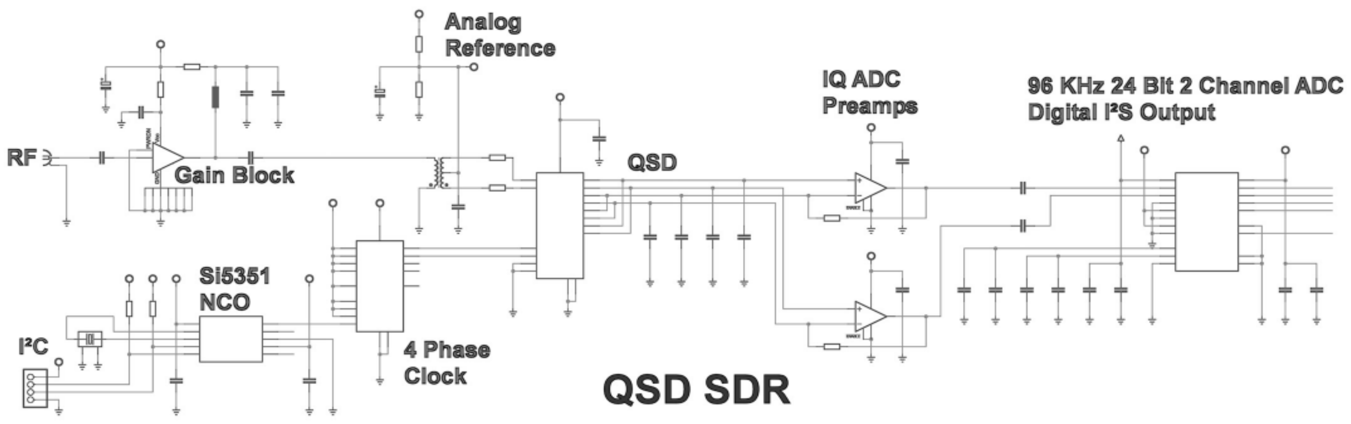
SDR, Arduino, Raspberry Pi

1. Introduction

The original impetus for this project was to design a replacement for a desktop general coverage HF receiver built by a well-known radio manufacturing company. Rather than trying to update the previous superhet design it was more direct for me, a embedded hardware and firmware engineer, to use digital rather than all analog techniques to realize the new design. It was also desired to support various digital modes directly within the new radio. A further departure from a traditional radio design was in not using a large number of mechanical user interfaces for control in favor of more graphical touch display user interfaces.

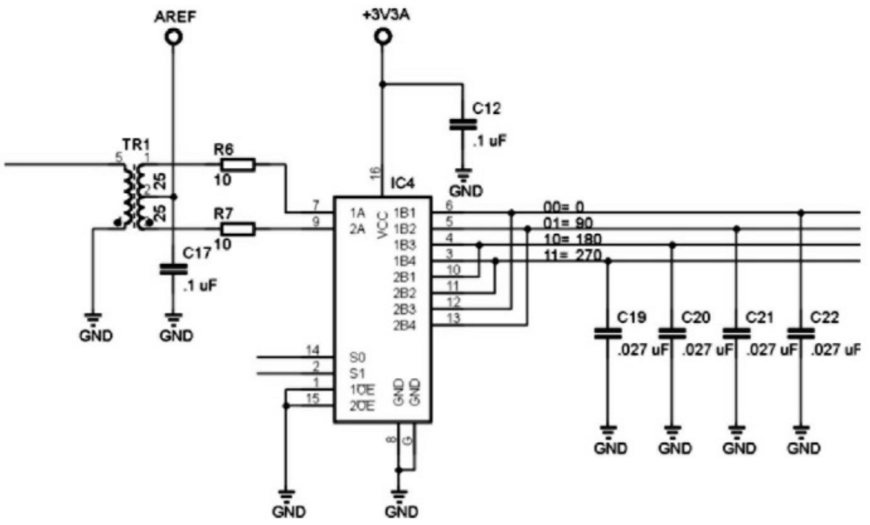
2. System Overview

A great deal of literature is available on the design of an SDR using a direct conversion quadrature sampling detector (QSD) topology and was referenced. Please see these for further explanations of the QSD SDR techniques. The design choices made in this implementation are describe here.



This design uses a double balanced QSD for low dB loss through the mixer.[1] A 1:1 ratio center tap .3 to 300 MHz transformer is used at the input on the SDR board. RF input to the SDR board is through a SMA connector. Front end switched filter banks required to reject residual images due to amplitude and phase imbalances in the mixer are external to the SDR board and controlled by the radio's processor.

2.1 A QSD is a two or more channel sample and hold circuit that is used to process an RF signal into frequency shifted analytical components. Mathematically the two analytical components, I and Q, represent the input signal on both the real and imaginary planes. Electrically the two output signals are an input signal that is shifted down in frequency and separated 90° in phase. Using the IQ signal format it is easier to process the signal information content digitally. Ideally mixing can take place with no images.



Each sample and hold circuit is built using an analog switch and capacitor. When the switch is connected to a input signal its instantaneous voltage is integrated with a capacitor. This circuit is arranged so that four capacitors are charged in turn covering 90° of signal each, that is 0°, 90°, 180°, and 270° at the clocking frequency. A numerically controlled square wave oscillator is used to clock a counter to generate the two bit switch sampling control sequence.

1 "Ultra Low Noise, High Performance, Zero IF Quadrature Product Detector and Preamplifier", Dan Tayloe

Signal pairs from the capacitors, 0° and 180°, and 90° and 270°, are combined by subtraction in two operational amplifier buffers to produce the IQ signals. The bandwidth of the received IQ signals before the opamps are determined by the charge time constant of the sampling capacitors and the input impedance through the RF chain. The cutoff frequency $Fc = 1 / (2\Pi \cdot Z \cdot C)$ where the impedance (Z) is a combination of the input impedance, the RF transformer turns ratio and differential splitting, internal on switch resistance of the analog switches, the series swamping resistors (used to reduce the percent variation of the analog on switch resistance), and a factor of four multiplier effects due to each charging capacitor being driven for only ¼ of a cycle. The voltage gain in the 1:1 differential RF transformer is 0.5, and the gain in the opamps is by their feedback resistance divided by Z times 2. Note that the opamps used in this design require a low input impedance for the best noise performance.

The analog to digital converters used in this design can run at a sample rate up to 96 KHz but are run here at 48 KHz. The ADCs actually sample at a much higher rate than the set sample rate, but include internal decimating digital bandpass filters to output the reduced sample rate. This greatly reduces the complexity of the antialiasing filtering required before the ADCs. The output of the ADCs are two channels of data encoded into a single 64 bit per frame I^2S encoded stream to the processor.

RF gain is realized in three sections: in a fixed RF gain block in front of the QSD mixer, in the low noise opamps used to drive the analog to digital converters, and inherent in large dynamic range of the ADCs themselves. The first stage of gain is made using a TI 12 dB fixed gain block amp. The TI TRF37A73 chip is designed to operate to gigahertz frequencies, but is tuned here to operate over the entire MF to HF spectrum. With the current values used in this design the gain through the RF transformer, QSD, and opamps, calculates to 35 dB for a total gain of 47 dB to the ADC.

The ADC used here has two synchronized channels of 24 bits each with a SNR of 103 dB. The reference voltage is 3 Volts p-p. With the front end gain of 47 dB the ADC overloads with a S9+45 signal present in the receive bandwidth. The minimum detectable signal is about .07 µv, although a more usable signal 2 bits above the noise would be .5 µv.

3. Hardware Description

For this project both an Arduino and a Raspberry Pi based single board computer were chosen to process the digitized IQ data generated by the SDR radio interface boards into received audio. An embedded design using any one of a number of processors could have been selected for the SDR application, but these two platforms are well known and offered powerful computing and interface options at minimal cost.

For the first prototype radio a Diligent chipKIT Wi-FIRE Arduino board was used. This board is WiFi enabled and has a Microchip PIC32MZEF MIPS processor. The PIC32MZEF runs at 200 MHz and has two megabytes of Flash program memory and 512 KB SRAM all internal to the chip. It features a DSP-enhanced core with four 64-bit accumulators, single-cycle MAC, saturating and fractional math, and hardware floating point. It has many channels of I^2S to support the digitized streamed IO needed. The audio amp used is a PmodAMP3 board, also made by Diligent, and is external to the SDR interface and Arduino board. It is based on an Analog Devices SSM2518 chip.

The SDR firmware controls two independent I^2S channels running at different sampling rates. One is used to capture the IQ data stream from the SDR interface board. This data, in fixed point format, is loaded into one of two memory buffers through a FIFO using an interrupt service routine. When a memory buffer is full a flag is set denoting data is ready to be processed, and the interrupt service routine then switches to the alternate buffer for the next input sample. A foreground routine processes all stored raw IQ data into decoded audio data switching buffers each pass. The processing time is short compared to the fill time of the buffers leaving plenty of processing time to perform other tasks. The second I^2S channel runs in a similar manner but is outputting audio data at a lower sample rate to a class D stereo amplifier driving two speakers. An I^2C interface is used to setup and control the stereo amp, the three channel numerically controlled oscillator (NCO), and to switch the RF front-end bandpass filter banks when required. The NCO is used to generate the clock for the QSD mixer, the ADC, and the audio amp.

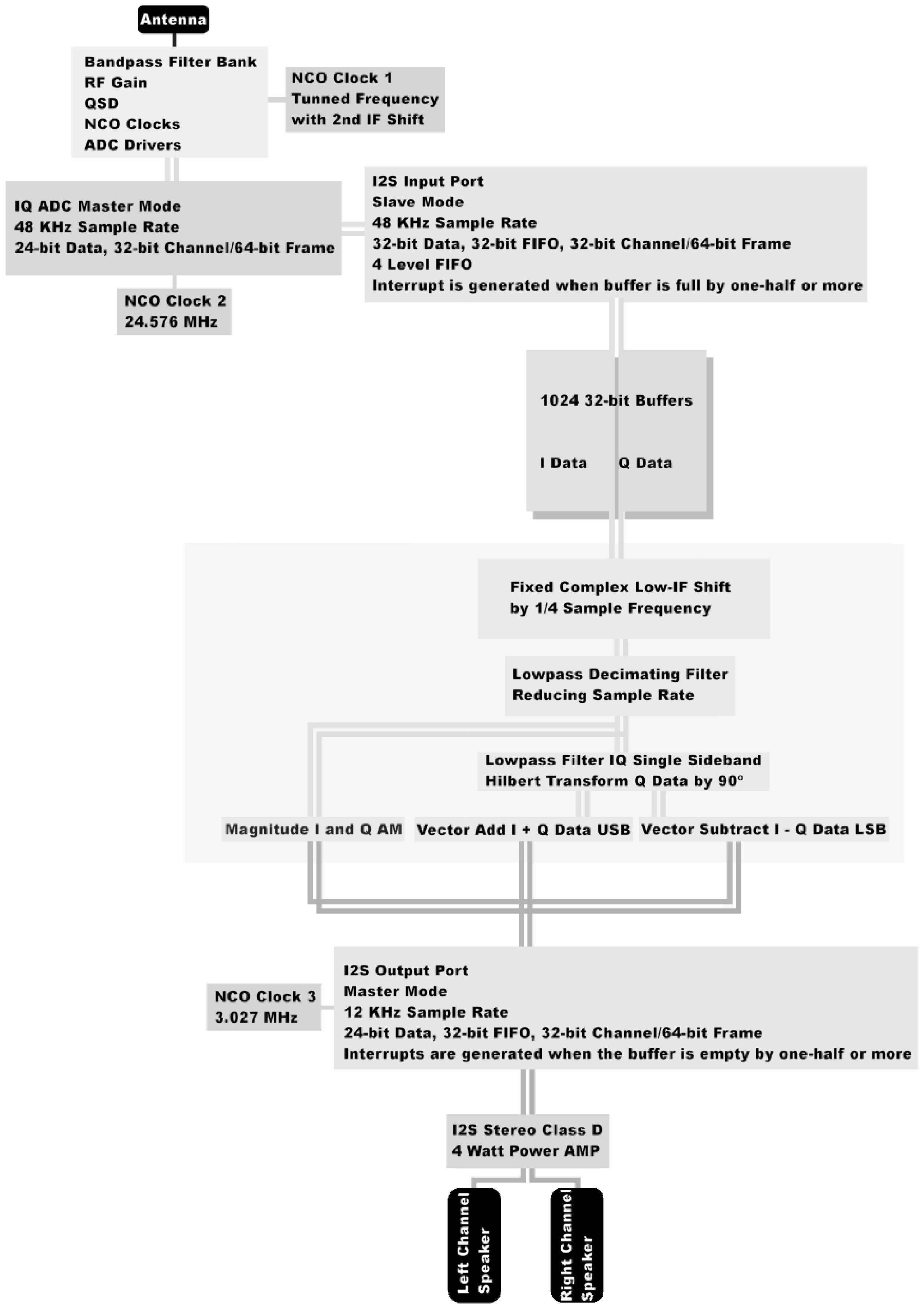
As a remote controlled server the full 48 KHz bandwidth of IQ data can be delivered with or without processing to an external computer, tablet, phone, or to the internet using Ethernet, USB or WiFi interfaces.

4. Firmware Description

The prototype radio firmware application supports only basic tuning and audio volume control, as in a simple desktop radio. AM, USB, LSB, and CW modes are supported.

All coding for the Arduino prototype was done using Microchip's MPLAB X IDE development environment in straight C and assembler under their Harmony framework. Microchip 16 and 32 bit libraries were called to implement the several fixed point finite impulse response (FIR) filters required to process the complex IQ data.

Initial testing was done on a Raspberry Pi running at just 700 MHz with C code written to implement the FIRs using calculations done by double precision floating point hardware. The ARM processor easily supported the required processing load.



Antenna
Bandpass Filter Bank
RF Gain
QSD
NCO Clocks
ADC Drivers
NCO Clock 1
Tunned Frequency
with 2nd IF Shift
IQ ADC Master Mode
48 KHz Sample Rate
24-bit Data, 32-bit Channel/64-bit Frame
NCO Clock 2
24.576 MHz
I2S Input Port
Slave Mode
48 KHz Sample Rate
32-bit Data, 32-bit FIFO, 32-bit Channel/64-bit Frame
4 Level FIFO
Interrupt is generated when buffer is full by one-half or more
1024 32-bit Buffers
I Data
Q Data
Fixed Complex Low-IF Shift
by 1/4 Sample Frequency
Lowpass Decimating Filter
Reducing Sample Rate
Lowpass Filter IQ Single Sideband
Hilbert Transform Q Data by 90°
Magnitude I and Q AM
Vector Add I + Q Data USB
Vector Subtract I - Q Data LSB
NCO Clock 3
3.027 MHz
I2S Output Port
Master Mode
12 KHz Sample Rate
24-bit Data, 32-bit FIFO, 32-bit Channel/64-bit Frame
Interrupts are generated when the buffer is empty by one-half or more
I2S Stereo Class D
4 Watt Power AMP
Left Channel Speaker
Right Channel Speaker

5. Radio Processing Algorithm

The above diagram documents the hardware and processing steps to realize this basic SDR radio using FIRs on the MIPS based Arduino board. It only requires a few changes to the procedure to be used with the Raspberry Pi platform.

The first stage of processing is a 2nd IF frequency shift by a small amount to tune within the baseband at an offset from center frequency. This is done to move away from system noise that leaks into the received signal. A complex sine cosine multiplication of the signal by a software numerical oscillator preforms the mixing without images. The firmware takes advantage of a special case and shifts the frequency by ¼ of the IQ sample rate to reduce the calculations required. At this rate the sines and cosines are always one, zero, or minus one and no trigonometric calculations are required.

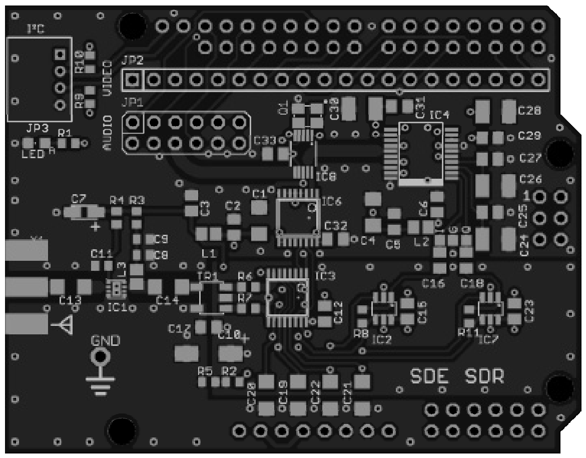
The next stage both filters and decimates the shifted IQ data stream. The decimation is set to reduce the sample rate from 48 to 12 KHz. This lower rate reduces the subsequent processing required, and the filtering sets the bandwidth to that used in AM mode. AM demodulation is simply the magnitude of the |IQ| data, where each audio data sample is the square root of the sum of the squares of the I and Q elements.

Single-sideband requires a further reduction in bandwidth and a Hilbert transform to shift the I and Q data 90° in phase between each other. Detecting the upper or lower sideband is determined by either adding or subtracting the transformed IQ vectors.

The bandpass characteristics in all the 88-tap FIR filters used can be easily changed in software by modifying the filters' coefficients.

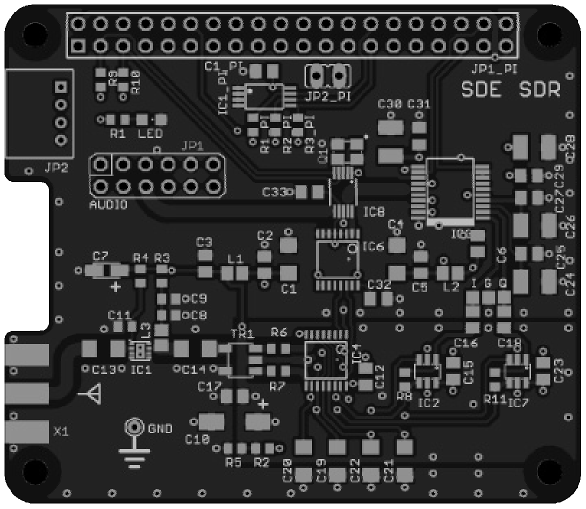
Several automatic gain control (AGC) algorithms have been tried for voice or data, all in software only.

3. SDR Arduino Implementation



A new four layer Arduino SDR "shield" has been designed that closely follows the running MIPS based prototype. It has a single ground plane and a split power plane for the various power supplies. Using the extended IO that is present on the Diligent Arduino the shield adds a provision for connecting an external color graphic touch screen as a user interface. A panadaptor and waterfall display can be supported. The audio makes use of a PMOD connector to route I²S audio data off the board to an external amplifier. The sound quality is good, and the design is usable for music, as well as voice.

4. SDR ARM Based Raspberry Pi Implementation



The Raspberry Pi version of the SDR is the same hardware design as used on the Arduino board. On the Pi “HAT” a video port isn't required. The Pi has but one I^{2}S port so that must be used for both data input and output with both running at the same 48 KHz sample rate. The extra processing at the higher sample rate is not a problem. The Raspberry Pi computers, although having less IO capabilities than embedded systems, have very fast processors and large resources. The Pi boards also benefit from having a full operating system. Program development for the board is currently being tested under both Linux, and Windows 10 using C# and C++.

5. Conclusion

Two platforms of a QSD Low-IF architecture SDR design have been developed for use as either a stand alone receiver, or as a remote controlled IQ data server. Future work will include support for digital modes for audio and data, including the reception of weak signals.

6. References

Doug Smith KF6DX, “Digital Signal Processing Technology: Essentials of the Communications Revolution” First Edition, ARRL, Newington, CT, 2003

Dan Tayloe. (2003, March). Ultra Low Noise, High Performance, Zero IF Quadrature Product Detector and Preamplifier. RF Design [online]. pp. 68-69. Available: http://rfdesign.com/images/archive/303Tayloe58.pdf

Gerald Youngblood K5SDR, A Software-Defined Radio for the Masses, Part 1, 2, 3 and 4,
QEX Jul/Aug 2002, pp. 13-21, QEX Sep/Oct 2002, pp. 10-18, QEX Nov/Dec 2002, pp. 27-36,
QEX, Mar/Apr 2003, pp. 20-31

Robert W Brown M0RZF, A Fresh Look at the QSD. [online] Available: http://www.m0rzf.co.uk

---

i 43 Greenfield Road, Milford, Connecticut 06460 System Design Engineering edkian@att.net http://www.cholakian.com Edward Cholakian has a BS in Engineering from the University of Connecticut and has worked in scientific programming, and embedded system development for 40 plus years.