A Different Type of Software Defined Radio – SDR Based on Labview

Software Defined Radio is a good example of HW & SW integration in Ham Radio. Labview, as a graphical programming environment, keeps away from a lot of program syntax. It frees up the programmer to focus on the system from a holistic point of view. This article shows the development of a SDR based on Labview and common HW-frontends.

Some years ago, I started with SDR and was excited about the capabilities and opportunities. With minimal investment I was able to work all types of modulation, having a very comfortable PC user interface. Frequency spectrums and their usage, image frequencies and modulation suddenly became visible.

One day my son called me to help program his LEGO NXT robot. Programming was done with a graphical SW called NXT–G, everything by drag and drop. NXT–G showed up to be a dramatically reduced version of National Instruments Labview. I was surprised and amazed and an idea started to form up in my brain to create a real time SDR with such a system.

Taking auto didactical lectures on Labview some major advantages showed up compared to a classical text oriented programming environment, like C. Many functions, from Graphics to DSP, are available by drag and drop. Syntax steps back, I was more and more able to focus on the overall system and its functionality. The created block diagram is the program itself, as it is very common to many HAM radio operators.

Front Panel

Picture 1 shows the front panel of the finalized SDR receiver with signals in the 40 m band.

Picture 1: SDR front panel
Major features are 2 independent receivers, a Softrock compatible SI570 frequency control, demodulation of SSB and AM signals and the implementation of different kind of filters.

The program is able to use internal and external soundcards from common SDR HW frontends.

The following chapters show how it works and how the program can be turned into action.

**Operating Mode**

Picture 2 shows the Labview block diagram of the actual RX version 1.2.1. It seems to be overwhelming, but notice. This is the whole program. It has to be compared to thousands of lines of code in a standard text oriented program like C. You can easily identify the different functional blocks.

![Labview Block Diagram](image)

Picture 2: Labview Block Diagram

Picture 3 shows the base building blocks of the SDR program.
I / Q signals from a Softrock compatible HW (e.g. FIFI-SDR, Softrock RX/TX, Funkamateur SDR) get processed block by block. Reading and writing of these signals takes place in different program threats. This enables a smooth processing with good performance. The whole processing takes place based on Complex Numbers.

Processing digital signals block by block and doing some changes on the signals (DSP) can create major distortions to the signals. Therefore an Overlap/Add procedure is implemented. Learn more about this secrets of DSP in [1] and [10].

Following a simple pre amplifier (just multiplication of the signals with a number), I / Q signals are transformed into the frequency domain via FFT. This also allows displaying the signals in a spectrum display.

Most of the following DSP routines work in the frequency domain (block made of dashed line). This is a big difference compared to other SDR implementations. The reason to do so is higher program performance as nearly all filtering is done via multiplication of transformed filter kernels with the signal. The filter kernels themselves will only be recalculated in case of any changes.

To enable split operation on different frequencies I have implemented a simple second receiver line without any complex filtering. Please note the second line in the block diagram.

The first processing of the signal in the frequency domain is band pass filtering. This isolates the target signal (spectrum) and later it is shifted into the base band.

In the main receiver line there is an unusual kind of noise blocker (squelch). It works with
spectral subtraction. If you control it carefully an overlapping noise is killed with very high efficiency.

As an alternative to that filter there is also a classical moving average noise reduction filter. Next block is a notch filter which is adjustable in terms of frequency range and target frequency. For easier use the frequency response of that filter is shown in a separate window, overlaying the selected target frequency range.

The iFFT is used to transform the remaining signal back to the time domain. The second part of overlap/add implementation closes this part of the processing.

The demodulation of the signal itself depends on the selected modulation type.

For AM the demodulation is nothing more than building the absolute value / magnitude of the complex signal. A constant current part is eliminated by a following filter.

For SSB a new way has been used. Normally either I or Q is shifted in phase by a Hilbert transform and than I & Q are subtracted or added from each other, depending on LSB or USB modulation type.

Here SSB demodulation is completely done in the frequency domain within the dashed line block. The background is a single sideband selection via the described band pass filter and a shift to base band. Using symmetry in the FFT / iFFT processing the Real part of the Complex time domain signal equals the demodulated signal. A suppression of the image frequency takes place anyway, again using symmetry in the FFT processing. The details of this DSP processing would go beyond the scope of this article. It could be called ‘FFT SSB Demodulation’.

In the main receiver line the next block is AGC processing with variable hang time. During the design of the program I found out that AGC is rather art than science as it is a total non linear process. Don’t expect an easy answer to ‘How can I ...?’ in this context.

Next, the demodulated audio signals are displayed and can be mixed to the left and right channels in multiple configurations.

Last functional block in the diagram is the output processing by block to the soundcard.

As a separate block there is also a SI570 frequency control via USB implemented. The SI570 chip is used widely in SDR HW to control the local oscillator.

The new version RX 1.2.1 also includes I/Q manual imbalance correction, I/Q swap, shutdown sequences to avoid and clear windows audio device errors and saving of parameters in a XML file.

In addition there is a version RX/TX 1.3.1 which allows SSB transmitting, SI570 PTT and

Labview SDR 4 16.08.2012
tune functions.

These new functions are not shown in the above block diagram.

**Installation**

Start with the RX version 1.2.1! If everything works fine you can always go to the RX TX version 1.3.1 later.

A compiled EXE version of the program for Windows is available. This requires also the Labview Runtime-Engine (RTE).

To use the SI570 frequency control 2 additional DLLs are needed: SRDLL.dll from [11] and a USB DLL from the SDR hardware. FIFI-SDR and Softrock RX/TX with SI570 control from DG8SAQ have been tested successfully with libusb0.dll. Both are included in the zipped archive.

So there are 3 simple steps to go live:

1. Download and install the Labview RTE 2011 from [12].
2. Download the EXE archive from [1] and unzip it to a location on your hard drive. Don’t change the structure or content of the folder.
3. Start the program by selecting the EXE file in the folder.

**Usage**

The usage of the program is simple as all parameters are set with meaningful values. All filters are turned off.

There is an important remark about stopping the program. This has to be done via the shutdown button. If you do it differently Windows will not release the allocated audio devices. Worst case Windows has to be restarted. I build in some security to not run into that situation, but users are very clever in finding out other ways...

The white arrow in the tools menu is used to start the program again.

Soundformat, Soundbuffer and Dev In/Out should be changed while the program is running but they will only become active after shutdown and restart of the program as all parameters are stored in an XML file.

Important parameter is the sample rate. It defines the spectral width you can work on the display. 44.1 KHz does always work as it normally is the soundcard default. Even if the soundcard supports higher sample rates that does not mean that it always works. Some soundcards do not switch successfully to another sample rate. The background is complex and goes back to the design of the Windows sound subsystem. Some soundcards come along with small programs which allow changing the sample rate, e. g. EMU 0202 USB. Use that program before starting the SDR.
For RX change DEV In/Out to the incoming I/Q signals and the speaker. For RX TX, a second soundcard is needed. The simple one located on the mainboard is fine for the TX channels. For decoding of DRM, PSK etc. a program called VAC has to be used to route the channels to another decoding program. Please note that VAC 4.10 has some issues, VAC 4.09 works fine.

All other parameters can be changed during operation and do not require a program restart.

Right after starting the program there should be stations, noise etc. in the spectral display. If not, change the level of the pre-amplifier or the SI570 frequency (also one time change after starting the program).

The stations can be selected by using the slider below the spectral display. Using the right modulation scheme and bandwidth a demodulated signal should be hearable from the speakers.

If there are distortions, try to reset the audio system with the corresponding button.

The notch filter can be adjusted with the 2 available sliders. Just try it, it is easy!

No needed explanation about the noise reduction filter. The noise blocker instead needs some attention. After switching it on there is a slider to adjust the level of spectral subtraction. This level is shown by a red line in the spectral display. A too high level causes distortions. The right level will effectively kill the noise.

With LO & Out Select there are 4 options to mix the output channels. This will also turn on and off the second receiver line.

With versions 1.2.1 and 1.3.1 there are some additional parameters for correcting I/Q imbalance for phase and level differences.

Another important new parameter is the display refresh rate. DSP and display was put to separate threats. The display processing is the single most performance consuming routine in the overall implementation. Change the refresh rate to your needs while looking the CPU power in the Windows task manager!

For TX the tune frequency is adjustable.

Please also note that the USB parameters in the SI570 section need to fit to your HW. The RX version 1.2.1 has the FIFI-SDR, the RX TX 1.3.1 version the Softrock RX TX with DG8SAQ interface set as a default.

**Experience and Perspective**

The Labview SDR program was extensively tested, mostly using SSB and AM on 40m, 80m,
and 20m with a FIFI-SDR and a Softrock RX/TX 6.2 with SI570 oscillator. Antenna used was an inverted-V dipole.

Digital modulation like PSK 31 and DRM with the DREAM software was also successfully tested.

Picture 4 shows the used SDR hardware, from left to right: German Funkamateure SDR [2], FIFI-SDR and a Softrock 6.2 RX TX with SI 570 oscillator.

![Picture 4](image)

Picture 4: Selection of SDR hardware

It is amazing which results can be achieved with such a software implementation. Without any formal measurement, a mid price analog Ham radio equipment was outperformed.

The design and the development of the Labview SDR provided a lot of fun to me. Maybe in the future I will implement something like direct PSK 31 coding and decoding, or a cross platform audio api like portaudio.

It was never planned to build something commercial or even reach the established programs with years of development like Power-SDR.

Implicit goal was to proof that full SSB modulation and demodulation can be done completely in the frequency domain. Having this goal, I learned worlds about DSP.

If you would like to get more details on the Labview SDR, please have a look at [1], for Labview also [3] and [4]. From my webpage [1] there is also the Labview source code (VI)
available for download. Unfortunately I do not have the time to translate everything from German to English, but the source code is documented in English.

For more information on digital signal processing please look at [10], [9] and [8]. A special focus to SDR DSP is described in [7], [5] and [6]. Most of it is in English language.

Fell free to contact me in case of any questions. I would also be interested in your experience with the program.

Have fun!

Michael Knitter, DG5MK
References


