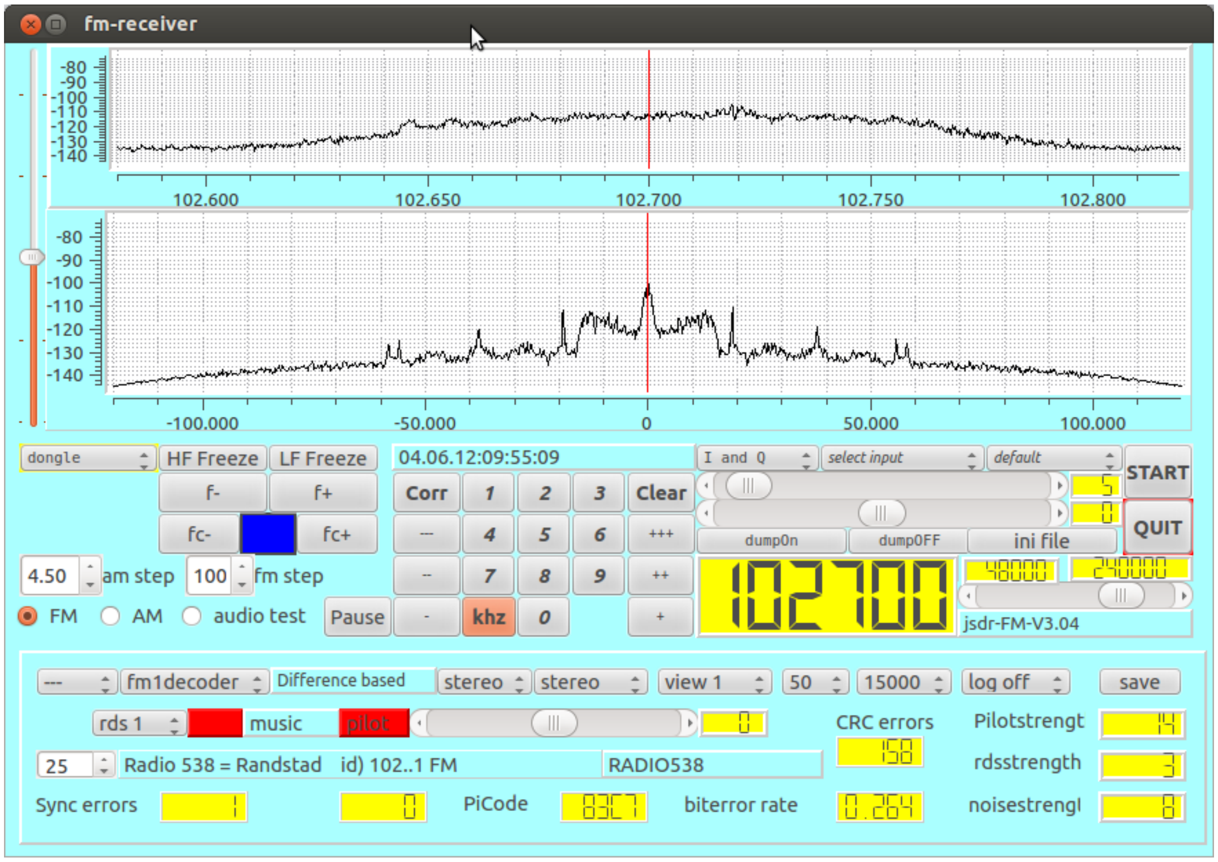
# SDR-J-4.1 SDR
*A software package for cheap sdr reception for Windows and Linux*

Jan van Katwijk
JFF Consultancy
The Netherlands
*J.vanKatwijk@gmail.com*



# 1 Introduction

SDR-J is the name of a set of support programs for software defined radio. The programs implement software backends for sdr receivers, covering with suitable - but cheap - hardware a range of 100K .. 1700 MHz for a variety of decoder modes.

The set developed from a very experimental program for supporting the Elektor SDR card (described in the Elektor May 2007, extension with preselector described in Elektor December 2009) into three backends supporting the pmsdr kit, rtl2832u based dab sticks and various other kits for which Winrad-conforming support is available.

To support the wider wider range of frequencies (55 Mhz direct and up to 165 Mhz through the 3-d harmonic of the Si570 oscillator) in the pmSDR kit, a separate version of the software, optimized for AM/FM reception, was derived. Recently, a cheap DAB stick was acquired, and using driver and control software, made available through the osmocom.org site for rtl2832u based DAB sticks, both the sw receiver and the fm receiver were extended with support for such sticks.

A spectrum viewer for use with the DAB stick was developed by stripping the FM receiver, such that up to a 3 Mhz wide spectrum of a user selectable band in the frequency range 50 .. 1700 Mhz can be shown.

The DAB stick can indeed be used for sdr purposes: its tuning range is wide, its frequency can be set and it delivers a stream of I and Q samples (8 bit though) through an USB-2.0 port. The DAB stick generates a stream with a data rate between 960000 and 3200000 samples/second (with a bandwidth accordingly). While the spectrum viewer needs the full bandwidth, decoding a WFM signal is better served with a much smaller band and - at least for reasons of performance - a lower sample rate. The FM software provides in decimation and filtering such that a user-selectable samplerate between 192000 and app 432000 samples/second is available for decoding results. Using 240000 or 288000 samples/second, it is quite well possible to decode RDS in the WFM signal (see picture above). The SW receiver provides in decimation and filtering such that samplerates of 48000, 96000 and 192000 are supported.

Since the Elektor card was - at the time - priced app 100 Euro, the pmSDR kit app 250 Euro and the DAB stick less than 30 Euro, the total cost of the complete set of hardware was less than 400 Euros, cheap and less than the laptop on which the software was developed.

The software is written in C++ in a Linux environment (recent versions of Fedora and Ubuntu), using *Qt* and *Qwt* for the gui. Soundcard support is obtained by using *portaudio*, an excellent portability layer. Support for reading and writing ".wav" files is obtained with *libsndfile* and *libsamplerate* for samplerate conversion. Finally, usb support is through *libusb-1.x*.

All these libraries are available using GPL style licenses for both the Linux and the Windows environment. Thanks to the availability of an excellent cross compilation facility, i.e. the Mingw64 cross compilation facility, on Fedora-17 generating an executable for windows (with statically linked libraries), turned out to be surprisingly simple. Software therefore runs both on 64 bit Linux as Windows, both 32 bit and 64 bit versions.

The software even runs on a 7 year old 1500MHz celeron based laptop, however, all compilation-time parameters are set to minimize the resource usage of the program, noticeable in the graphics for displaying spectra. Resource consumption, however, is pretty high, and on such very old machines FM stereo tends to stutter. It certainly runs very fine on modern 64 bit systems, both Windows and Linux.

The software is being developed as a hobby project and all sources are available under an Open Source License, the GPL V2 or V3. It's licensing is constrained by the licenses of the many open source libraries Apart from libraries used, other existing software available under similar open source licenses, e.g. gmfsk, fldigi, cuteSDR, served as source of inspiration and provided some lines of code for the various parts.)

# 2 Basic usage of the programs

Since the three programs originate from the same source, their "look and feel" is the same. Indeed, many underlying support modules and mechanisms are the same, and all three programs are based on Qt and Qwt as vehicles for creating GUI's.

## 2.1 Program initialization

All three programs are such that on startup, reasonable defaults are used for the setting of the switches and sliders. The three programs share a common mechanism for initialization through an *ini* file. Such an ini file contains settings for the initial value for selected sliders, buttons and switches. Obviously, these files are ASCII files and can be adapted manually (e.g. the looping range). Default place for ini files is the $(HOME) environment. The filenames are *.jsdr-fm.ini*, *.jsdr-sw.ini* and *jsdr-spectrum.ini*

The SW receiver will save the settings in an ini file on (normal) program termination.

Usage of ini files can be influences by setting switched from the command line.

### 2.1.1 The "-d" switch

A *new* ini file is generated easily by running the program with the *-d* option: at (normal) program termination the settings are recorded in the default ini file.

### 2.1.2 The "-i" switch

Applying the *-i filename* option on the command line, provides the opportunity to use another file than the default ini file.

#### 2.1.3 The "-o" switch

The *-o filename* option provides the opportunity of dumping the status of the various sliders and controls into an ini file at program termination.

#### 2.1.4 The "-e" switch

The *-e* switch is applicable for the SW receiver. It tells the software *not* to store the configuration information at program termination.

### 2.2 Selecting soundcard, devices and starting the program

When starting the execution of the program, the actual processing of data will not start until the *start* button is pressed. However, when starting, appropriate devices for generating sample-streams in and accepting sample-streams out should be available. For the *spectrum viewer*, the options are limited, the program assumes to be connected with an RTL2832U based dongle while there is no output-stream. Obviously, settings may have been influenced by the settings in the ini file, there are actual choices, however, for the SW receiver and the FM receiver. No "defaults" are selected for soundcards. Obviously, selecting a soundcard (and a radio device) can be done through the ini file.

The selected "radio device" can be:

- *no rig*, the program will not control any external device, but it will just process the data coming into the selected soundcard-in.
- *pmsdr*, the program will assume that a pmSDR kit is connected, it will open the usb connection to that kit and display - on success - a small subwindow for selecting - either manually or automatically - a bandfilter within the kit. Data will be read - as with the *no rig option* - from the selected soundcard.
- Selecting *dabstick* or *dabstick (dll)*, the program will assume that a dabstick is connected and will (try to) open the usb port to that dab stick. Input will be arranged through the dab stick interface.
- Selecting *replay*, the program will open a selection window for a (wav) file to be selected as input device. Note that processing data from an input file is endless: om encountering the the end of the file, the file is reread. Notice furthermore that the rate of the samples in the *.wav* file will be adapted to the selected samplerate.
- Selecting *extio*, the program will open a selection window for an Extioxxx.dll file to be selected as driver for the attached hardware. *This facility is still experimental and under development.* The facility has been tested for the dll's for the elektor card, the pmSDR kit and dabsticks (i.e. my own hardware). Note however, that since these dll's are 32 bits, the facility is essentially limited to the Windows 32 bits version. Trying to open a 32 bits dll file from within an x64 executable

will fail. The facility as such, however, is supported in bot Linux and Windows. Note furthermore that the facility in its current form supports - apart from the aforementioned dab stick - dll's for devices that are will send the data to the soundcard and conform to the specification of Alberto for dll's (type 4 hardware).

In all cases an appropriate selection should be made for *output* through a soundcard. The choices for selection of soundcards follow from the local OS and the soundcard(s) connected to the system. For windows, there is support for wmme, directx and asio, for Linux there is support for devices that are supported by Alsa.

Note finally that these observations do not hold for the spectrum viewer, this latter program does not generate output and its input is coming from the dabstick, which is automatically selected as radio device.

## 2.3 Handling Frequencies

The programs share the interface for entering and modifying frequencies:

- *Keypad.* Through the keypad a frequency can be entered, for the sw receiver the frequency can be specified in either Hz or KHz, for the other two specification is in KHz. Entering an invalid frequency will - in general - have no effect (Notice that the allowed frequency range for DAB sticks may vary, for DAB sticks only a rather general test on the legality of the frequency is done.)

- keys for shifting the frequency small amounts, are next to the keypad,

- *Stepping.* The FM receiver and the spectrum viewer provide support for easy manual and automatic stepping through a range of frequencies. The $f++$ and $f--$ buttons support manual stepping, $fc++$ and $fc--$ allow automatic stepping. Touching the $fc++$ or $fc--$ button more than once will alter the stepping period. Touching the $fc++$ when automatic stepping is going on with a negative step size, will decrement the step time. Similarly for $fc--$, which then decrements the step time for the positive direction.

- *mouse click selection* The SW receiver supports selecting frequencies by clicking on any of the two displays, the FM receiver supports selecting a frequency by a mouse click on the top display[1], as does the spectrum viewer (which only has a single display).

- *Mousewheel rotation.* Starting with version 4.1 turning the mousewheel will also affect the selected frequency with small steps (0.1 KHz for the SW receiver, 1 KHz for the FM receiver and 50 KHz for the spectrum viewer.)

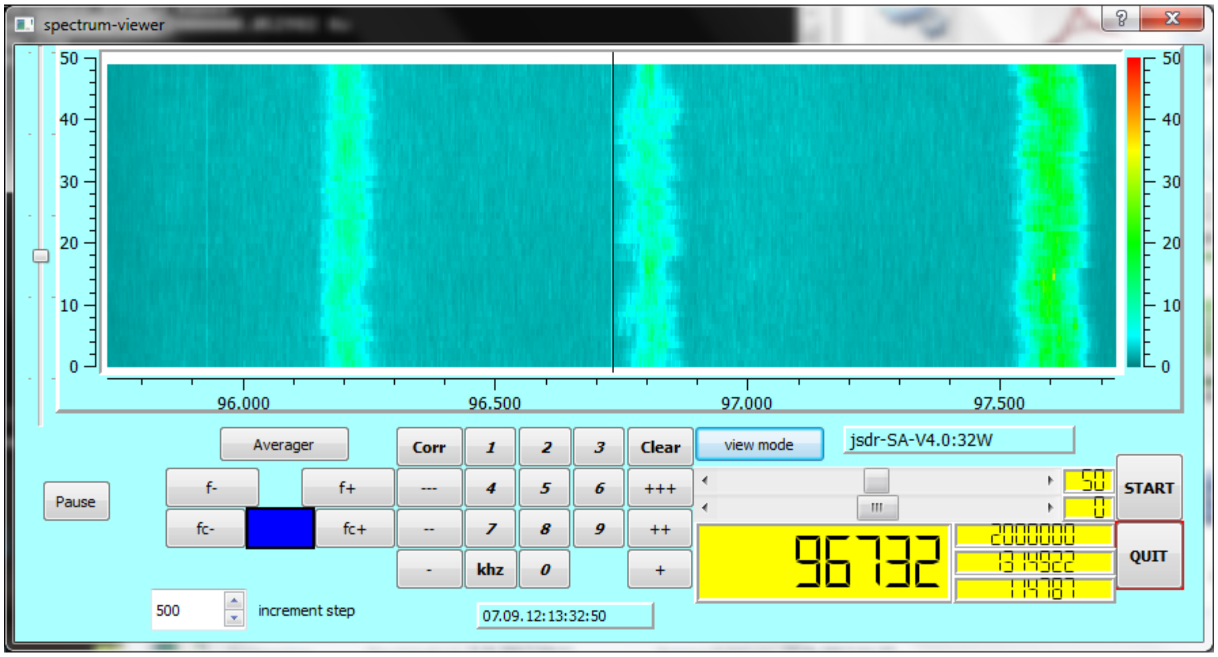
[1]In the x64 version of the FM receiver, clicking on the bottom display will cause some zooming in into the display of the frequency spectrum

# 3 The programs

## 3.1 The spectrum viewer

The spectrum viewer supports DAB sticks with the rtl2832u and any of the tuners fc0013, fc0012, e4000, the fc2580 or the R820T. Software support for the stick and tuner is implemented through a simple wrapper around the available osmocom rtlsdr.dll.

The speed of the incoming datastream - and therefore the spectrum width - with the -*C XXX* command line switch, where XXX is in Hz. The default setting is 2MHz, selection between 9600000 and 3200000 Hz is possible. The spectrum viewer will show the spectrum of the incoming samplestream.



The figure shows a spectrum of a part of the FM broadcast band, with a spectrum width of 3 MHz (with just the "antenna" that is coming with the stick).

The actual frequency can be set as described previously: through the keyboard and altered by either typing a new frequency, or by stepping though the frequencies, by turning the mouse wheel, by manually (f++ or f– button) or automatically (fc++ or fc– button). Step size as well as step time can be set, The default step size is set to 500K, but can be set differently. Scanning the full FM broadcast band with a step size of 100K, and a step time of 1 second will take a couple of minutes. *Boundaries* for stepping (such that stepping will be repeated when the upper or lower boundary is reached, depending on the step direction) can be set in the ini file, the built-in default values are 4500 KHz to 1700000 Khz.

## 3.2 The SW receiver

The first program that was developed in the set was (a predecessor of the current) SW receiver. The program is still the most complex one of the three (complex from the user's point of view): it contains by far the most selectable items. It was originally built with the Elektor SDR card in mind, i.e. for frequencies from app 150 KHz to 30 MHz. Later on, it was extended with support for the pmsdr kit. *When preparing the software for cross-compilation, direct support for the Elektor card was lost. Starting with version 4.0 support for this card is made available through the* extio *selector using an appropriate dll.*

Support for the pmSDR kit is two ways. There is direct support, and through the *extio* selector using an appropriate dll. Difference is obviously that the 64 bits Windows version as well as the Linux version are unable to process the 32-bits dll, and only provide direct support.

With the pmSDR kit we get a receiver with a continuous tuning range from 100KHz to 165MHz. When tuning to a frequency in the high bands, the software itself will take care of transformations to be done, i.e. switching to the 3-d harmonic and switching the I an Q.

Finally, support for the above mentioned DAB Stick was added. With the DAB stick, we get a receiver with a continuous tuning range from 45MHz to 900 (sometimes 28MHz to 1700 MHz).

When connected to the pmSDR, input samples will be read though an attached soundcard, and - obviously - a card, an input device, should be selected. When connected to the DAB stick, input samples are coming though the USB bus.

Default inputrate for soundcard devices is 96K, this can be set to 192K (-T option) or 48 K (-S option) through a command line parameter. Standard outputrate is 48K.

When reading from the DAB stick, sampling speed for the DAB stick can be set, default value is 960000 Samples/second. These samples are filtered and decimated to the selected inputrate. To allow an easy use of the DAB stick in combination with the SW receiver as a backend for an upconverter, starting with version 4.1 an *offset* can be specified. This offset will be added to the selected VFO for the DAB stick.

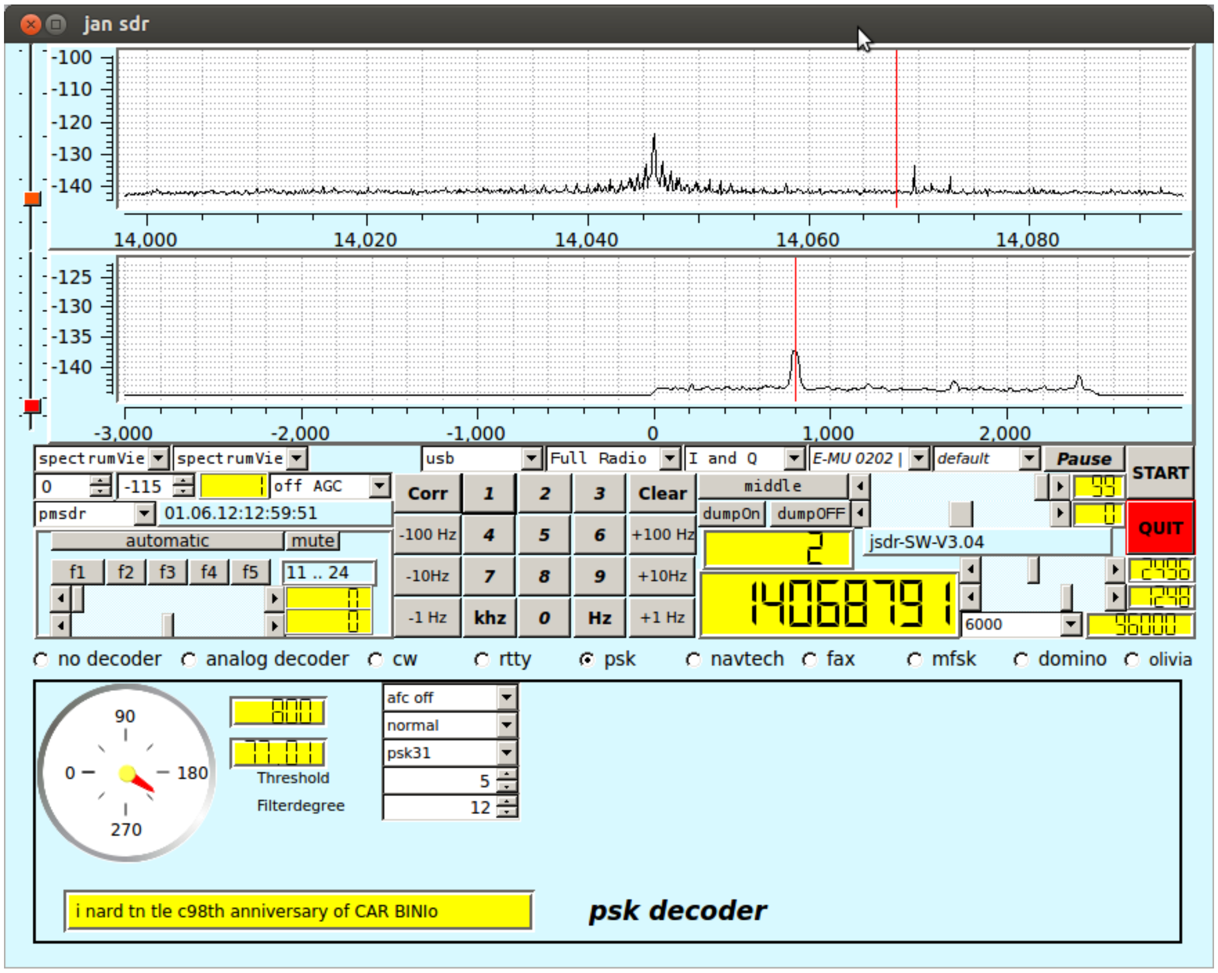
The receiver has two displays, one showing the spectrum of the input samples, the second one showing the spectrum of the filtered, shifted and decimated sample-stream, i.e. the stream entering the selected detector. Clicking on a point on any of the displays will adjust the input frequency to where is pointed to. Frequencies can be specified with an accuracy of 1 Hz.

The Gui contains a switch for each of the displays to be set to show a waterfall rather than a regular spectrum. Obviously, preference can be recorded in the ini file.

A *bandfilter* can be set around the specified frequency and the selected band then will be decimated to the *working rate*, i.e. the sample rate on which the decoder operates. Both the bandwidth and the center-frequency of the filter can be manipulated by sliders on the gui.

The filter is an FFT implementation of a FIR filter (it actually consists of two filters) with an effective order over 1000. Some standard filter settings, e.g. for am usb, lsb, fax, are predefined and a selection can be made by a simple mouse click. Practical band selection is from app 100 Hz to the working rate.

One of a number of predefined working rates can be selected (4000, 6000, 8000, 96000, 12000).



i nard tn tle c98th anniversary of CAR BINIo

As shown in the above picture, the radio device used here is the pmSDR kit, somewhere on the 20 M band, with a simple wire as antenna.

A variety of decoders is available. For each of the decoders a separate subscreen (the bottom part of the gui screen) is shown, with selectors, sliders and buttons specific to the selected decoder. The screen will appear when selecting the decoder.

- As default, a ”no decoder”, is available. This decoder does essentially nothing: it just passes decoder input to the output, while showing the input samples on a small complex plane;
- An analog decoder is selectable. This decoder implements the the common analog

decoder modes such as am, fm and ssb. For fm decoding a squelch is available.

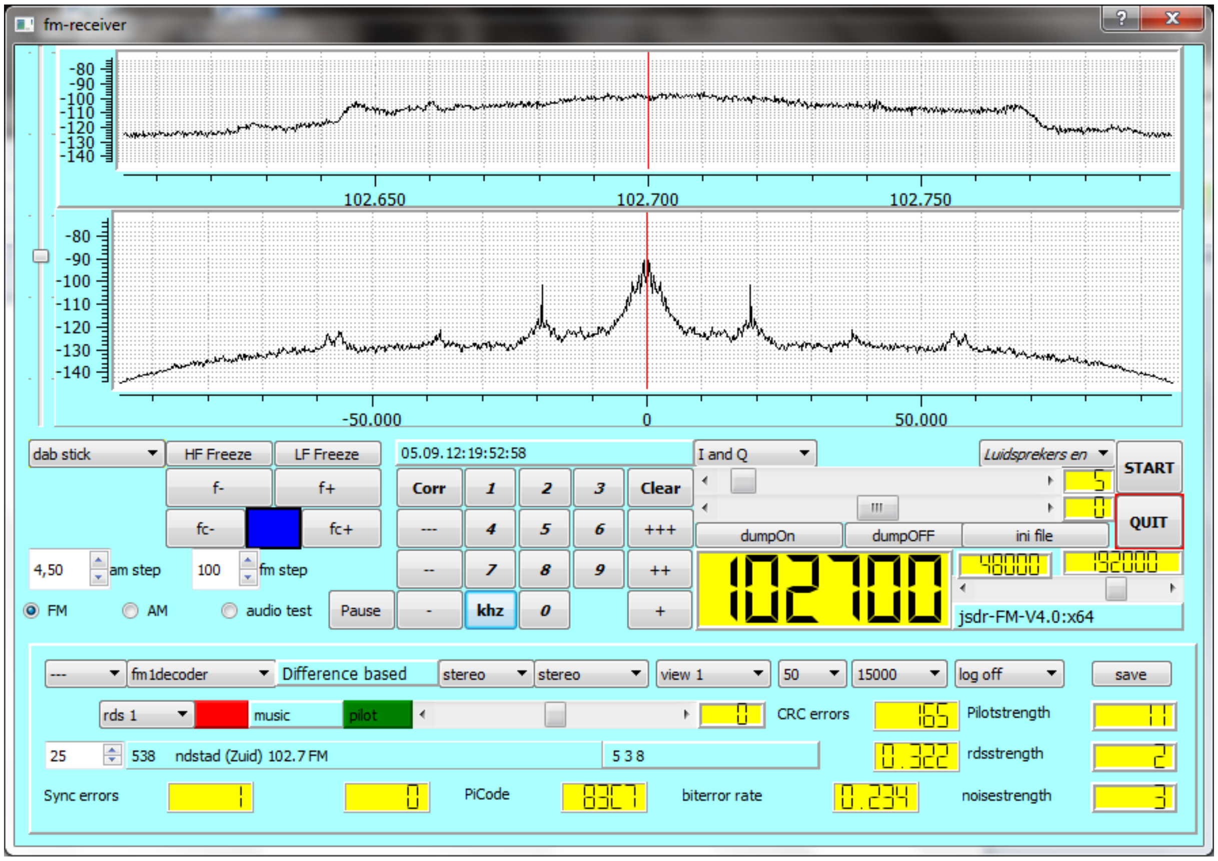
- An rtty decoder can be selected. This decoder implements rtty decoding with lots of user selectable parameter settings commonly found on rtty decoders.
- A CW decoder can be selected. This decoder implements *adaptive* CW decoding, again with lots of user selectable settings. The CW decoder attempts to guess the width of the dot and the dash, based on a setting for words per minute. Increasing the accompanying counter is useful for very rapid transmissions.
- A psk decoder can be selected. This decoder implements both bpsk and qpsk decoding for a variety of speeds.
- An mfsk decoder can be selected. This decoder implements the common mfsk modes, and is equipped with additional visual tuning aids. Mfsk should be tuned with an accuracy of a few Hz.
- A domino decoder can be selected. This decoder implements the common domino modes.
- An olivia decoder can be selected. This decoder, essentially a wrapper around a package developed by Pawel Jalocha, implements the standard olivia modes.
- A wheatherfax decoder can be selected. This decoder, a partial reimplementation of the work of Christoff Schmitt, is implementing wheatherfax decoding. The received picture can be stored into a file.
- A navtex decoder can be selected. This decoder allows coastguard and other amtor-B messages to be decoded and made visible.

Obviously, there is a provision for dumping the input samples into a ".wav" file, and processing these (or other) ".wav" files. When processing ".wav" files, sample rate conversion is supported, so files created during a session when running e.g. 192K are valid input when being processes during a session running e.g. 48K and vice versa.

Finally, the receiver provides the opportunity to *not* select a radio device, but to process data that is entering through the soundcard (the button now labelled "full radio"). It is then possible to just pass this data on (after decimation) to a decoder, allowing output from any other receiver to be processed as well. I am using this option with an old shortwave receiver.

## 3.3 The AM/FM receiver

The third program is dedicated to AM/FM reception. The program implementing the receiver supports - next to file input - the aforementioned pmsdr kit and the DAB stick as discussed above.



The software, when connected to the pmsdr kit, supports, again, a continuous tuning range from 100K to 165 MHz. The software, when connected to the dongle, supports a continuous tuning range from 45 MHz to 1700 MHz, depending on the tuner capabilities.

The frequencies below 30 MHz mostly provide AM broadcasts, the FM band obviously provides FM broadcasts.

As with the spectrum viewer, automatic (or manual) stepping through a predefined frequency range is supported, as is - obviously - manually setting whatever frequency in the domain accepted by the connected device. Automatic stepping range depends on the mode used. Default automated stepping in AM mode is only useful in combination with the pmsdr: frequencies for common AM bands are not valid when using the dongle. However, frequencies for stepping can be set in the ini file. Default automated stepping in FM mode covers the 86.5 to 108 MHz band, useful with both pmsdr and dongle. These frequencies can also be set in the ini file.

For each of the reception modes (AM and FM) a separate subwindow is defined with controls and displays specific to the selected mode.

For the AM it is simple and not shown here, it contains a selector for the passband that is to be demodulated, it contains a switch and setting for the AGC and a selector for any of a few different detector implementations (envelope, pll).

The focus in the implementation is on the FM receiver. A selection can be made from 5 different implementations of an FM demodulator. A choice between stereo/mono can be made. When selecting *stereo*, output can be set to: stereo, the left channel, the right channel, and the L+R part as well as the L-R part of the signal. In general, the quality of the L-R channel is indicative for the overall quality. The standard de-emphasis filters can be selected, and the width of a low pass filter is selectable.

RDS decoding is supported: a selection can be made between *no rds* decoding and 2 different decoding implementations.

Furthermore, an input bandfilter can be selected for a number of predefined widths and a selectable strengths. It must be noted, however, that running the software in a 32 bit version on older PC's, processing and demodulating FM, selecting a filter and decoding RDS may consume too many of the limited resources.

As an aid in understanding the demodulated signal, there is a choice in what the second display will show, the full demodulated signal or one of its constituents. As additional feature, a log can be made of some relevant variables.

What must be noted is that the samples coming from the DAB stick are 8 bit samples, resolution is therefore limited. However, as the picture shows, even then RDS decoding, at least partially, is possible for some of the stronger FM broadcast stations.

Some command line options are specific to parameter settings for the DAB stick:

- The *-E* option tells that internally 240000 samples/second will be handled rather than the default 192000. In the current settings, this is not useful for other devices than the DAB stick.

- The *-F* option tells that internally 288000 samples/second will be handled rather than the default 192000. In the current settings, this is not useful for other devices than the DAB stick. When set to 288000 samples/second, performance (as well as CPU usage) increases.

- The *-T number* option is specific to the DAB stick. The number will be multiplied with the outputrate to compute an inputrate. E.g. the -F is a shorthand for -T 6, the -E a shorthand for -T 5.

- The *-m number* option is specific to the DAB stick. It is used to map the selected internal sample rate to the rate of the DAB stick. The *number* should be even (for technical reasons). My preferred setting (on a 3 year old laptop) for command line options is -F (i.e. DAB samplerate 2880000, decoder rate 288000), although -F -m 8 (i.e. decoding rate 288000, DAB samplerate 8 * 288000) and -E -m 12 (i.e. decoding rate 240000, DAB samplerate 2880000) also give good results. The *-m number* is ignored and set to an acceptable value when application of the given number would lead to a request to the DAB stick for a samplerate outside the range of 960000 .. 3200000 samples/second. E.g. -T 10 -m 10 would normally lead to a request to the DAB stick to deliver 4800000 samples/second, which is refused.

The precompiled settings for the x64 version are *-m 10*, for the w32 version the default setting is *-m 6*. It will be obvious that an increased decoding rate will increase the amount of resources needed for demodulation and processing.

As with the SW receiver, the input samples can be written onto a (.wav) file, and wav files can be used as input for the receiver.

# 4 The distribution

## 4.1 The files and directories

The distribution consists of three parts. One part contains the full set of sources, the other two parts contain a pre-packed set of executables for windows-32 and windows-64 respectively. The packages with the executables contain the required dll's for execution.

Packing may differ from version to version, the sources are always packed as a ".tgz"file. When unpacked, a tree is generated with directories:

- *docs*. This directory contains this document.
- *ini-files*. This directory contains example ini files for the three programs. Note that their names when used as default ini files should start with a dot (.) (e.g. ".jsdr-sw.ini"). In order to use these files, please prepend a dot (.) to the name before copying them to "$(HOME)" under Linux and "tmp" within the local folder for Windows.
- *swreceiver* which contains the main sources for the sw receiver,
- *fmreceiver* which contains the main source needed for the fm receiver,
- *spectrum-viewer* which contains sources for building the spectrum viewer.
- *righandling*, which contains all files related to the radio devices,
- *scopes*, which contains the various files for the implementation of the displays,
- *filters*, which contains the implementation of the various FIR and IIR filters,
- *soundcard*, which contains the interface code to the portaudio library, as well as the file reading functions,
- *utilities*, which contains a few support functions, and
- *viterbi*, which contains an implementation of the viterbi algorithm, used in the implementation of qpsk and mfsk modes.

## 4.2 Building the programs

### 4.2.1 Building under Linux

Building the programs requires the availability of a number of libraries (i.e. library packages and library packages). To facilitate the building process a script is available that, when started, will attempt to load the required packages. For ubuntu the script is named *prepare-ubuntu*, for fedora the script is named *prepare-fedora*.

Having run this script successfully all prerequisites are available. Creating an executable is then

```
cd .../make-x64/XXX
#(where XXX is either fmreceiver, swreceiver or spectrum-viewer)
qmake (or qmake-qt4 when running fedora)
make
```

*Cross compiling* is possible, this requires the availability of the cross compilation suite and cross compilation of all packages. This distribution is built using the mingw cross compilation packages under Fedora 17.

### 4.2.2 Building under Windows

Using the Mingw environment and the Mingw/Qt distribution it is well possible to create an executable under windows. If we assume a full Mingw/Qt environment to be operational, all packages have to be compiled and installed using the gnu autotools, after which the building of any of the three programs is as under Linux. The main advantage of building under Windows could be that in that environment all libraries are available as "dll", such that libraries could be linked dynamically and the size of the executables is much smaller.

# 5 Sources of inspiration

The software uses many ideas - and in some cases lines of code - from others. The main sources of inspiration for various parts - other than Qt or Mingw - are

- *Signals, Samples and Stuff: A DSP Tutorial* by Doug Smits (QEX 1998, 4 parts), really the set of papers that made me decide to do some programming in the field of SDR.

- *The Scientist and Engineer's Guide to Digital Signal Processing* By Steven W. Smith, Ph.D. California Technical Publishing P.O. Box 502407 San Diego, CA 92150-2407. This book, found on the internet, provided me with a first introduction to digital processing. All 640 pages are available on the internet.

- *Practical Analog and Digital Filter Design* by Les Thede, Artech House, which provided me with the basics for IIR filters, their design and their implementation. I found this book on the net as well, and it is still an often read piece of work.
- *Implementation of FM Demodulator Algorithms on a High Performance Digital Signal Processor* by Franz Schnyder, Christoph Haller. Diploma Thesis Hochschule fur Technik Rapperswil, Nanyang technological University 2002. Gave a readable overview on 4 (5) different algorithms for the decoding of FM.
- *CuteSDR Technical Manual*, October 2011 by Moe Wheatly, which gives a good view on various techniques used in the cuteSDR software and helped to improve some of the algorithms used.
- sources of many existing programs, in particular
  - osmocom rtl-sdr, a set of programs and drivers for the rtl2832u based DAB sticks (see osmocom.org), which forms the basis for the interface to the stick.
  - RTTY, an FSK decoder program for Linux (Jesus Arias, 2001). This program gave the first hints to process CW and RTTY.
  - fldigi (W1HKJ), that gave the hint on how to handle CW with an adaptive algorithm.
  - gmfsk (Tomi Manninen OH2BNS). A great program that gave a real insight in decoding the psk, mfsk and domino.
  - hamfax (Gerhard Schmitt), which was the source for the fax implementation.
  - digiRadio (Alberti Perotti and others), and
  - FM Stack (Michael Feilen), that gave the inspiration for the implementation of FM software.
  - The pmsdr interface software (Martin Pernter, Andrea Montefusco), and the Si570 data sheets. The current pmsdr interface has some parts that are directly derived from this interface software, while the Si570 code is merely derived from various other sources on the internet.
  - The elektor interface software (Burkhard Kainka) and the hamlib software that learned me how to handle the Elektor card.
- and numerous anonymous sources on the internet.