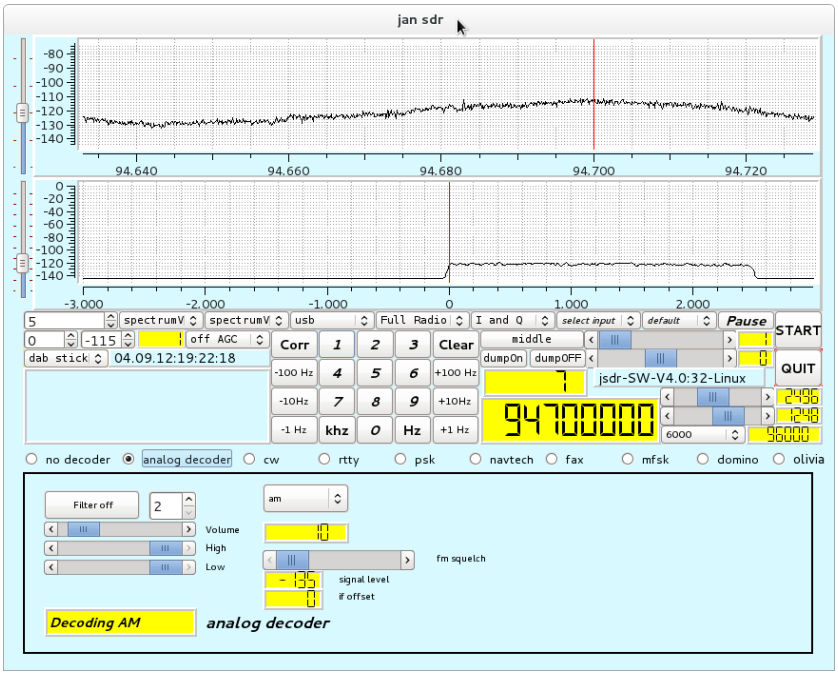
SDR-J-4.0

Software for a simple software defined radio

The shortwave receiver

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

September 4, 2012



1 Introduction

SDR-J (previously called jff-sdr JSDR) is an implementation of a set of support programs for software defined radio. The programs implement a basic receiver in the sdr domain: functioning, but improvements are possible and needed.

The set is written in C++, the programs were developed under Linux (recent versions of Fedora and Ubuntu, both 32 and 64 bits versions), using Qt (and Qwt) for the gui. Thanks to the portability of Qt and C++ and thanks to the Mingw cross compilation facilities, porting the software to Windows turned out to be surprisingly simple (though not trivial).

The software has been tested under Windows XP, Vista, W7, Fedora 12/13/14/15,16,17 and Ubuntu 10.04/11.10/12.04. Currently I am running Fedora-17 (64 bits) and Windows-7 as development resp. test platform.

The software is being developed as a hobby project and is available as open source under the LGPL V2 and V3. It builds upon many open source libraries (e.g. Qt, Qwt, fftw3, libusb-1.0, libftdi, libsamplerate, libsndfile, libportaudio) while other existing software available under open source licenses, e.g. gmfsk and fldigi, served as source of inspiration (and some lines of code) for the various decoders.

The complete set contains a shortwave receiver, an fm receiver and a spectrum viewer (swreceiver, fmreceiver resp. spectrumviewer). In this manual the sw receiver is discussed. This swreceiver implements:

- *Control*.
  - Support for controlling the basic-PMSDR[1] kit is included as is support for RTL2832U based DAB sticks.
  - Support for the use of existing "Extio*.dll"s (supporting sdr kits) is in limited form available. Existing dll's for e.g. pmsdr and the elektor kit are 32 bit and therefore this facility effectively works under the w32 version. The ExtIO_RTLSDR.dll that was developed for sdrsharp is also supported.
  - The Elektor plug-in supports continuous tuning between app 150K to 30M. The PMSDR supports - through the use of the 3-d harmonic trick - continuous tuning between 100K and 165MHz. For frequencies above 55 MHz, the 3-d harmonic is used and all kind of filtering is switched off.
  - The dabstick supports continuous tuning between app 45 MHz .. 1500 MHz.
- *Reading and processing* of (soundcard) data, filtering of the data, frequency transforms of the data into a baseband signal and displaying frequency information the the form of spectra.
- *Decoding* of the baseband signal:
  - analog signals for the modes AM, USB, LSB, ISB and (smallband) FM,
  - CW,
  - digital modes:
    - rtty,
    - psk,
    - navtex,
    - mfsk,
    - (wheather)fax,
    - domino,

[1] The 100K .. 55 Mc version is supported, the UHF extension card is not, simply because I do not own such an extension card.

* olivia.

Data may pass the front end stage, decimated but further unaltered, useful when another source for generating program input such as an existing shortwave receiver is used.

Furthermore, the software supports dumping the unaltered input samples into a ".wav" file, and reading ".wav" files back, processing their content.

Samplerates for input and output streams are frozen per run of the program. When starting the program from a command line, the samplerate can be set by the -S option (48000 samples), -T option (192000 samples) and -R option (96000 samples). The default samplerate is 96000 samples/second. OutputRate is 48000 samples/second in all cases. Internal operation is usually on a lower samplerate. The samplerate used internally can be changed dynamically.

General remarks The software most likely contains errors. Although the numbers of errors found was exponentially decaying for the past year, it is an illusion to assume the software to be error-free. All kind of feedback is welcome.

The pictures in this manual are herited from previous versions of the software and its manual. The screens on operating may differ slightly from the ones presented in this manual.

2 The distribution package

The distribution is in the form of two tarballs, one containing a 32 bit executable, the other containing the 64 bits executable, together with the required dll's and separately a tarball containing all sources and information to build the executables under Linux.

- *docs*, the directory containing a copy of this document.
- *filters*, the directory containing all software implementing the various filters being used (IIR, FIR, and fft-based FIR).
- *swreceiver*, a directory containing all files specific to the shortwave receiver. The main driver program is in the file called g̈ui.cpp; which is essentially interfacing the logic to the user interface.
- *fmreceiver*, a directory containing the source code for the FM software.
- *spectrum viewer*, the directory containing the implementation of a simple spectrum viewer for the noxon stick.
- *make-w32*, a directory containing the subdirectories with ".pro" - and makefiles for creating executables with 32bits floats, i.e. for the w32 versions, as well as a *jff-include.h*, an include file with some system wide definitions.
- *make-x64*, a directory containing the info for the 64 bits versions of the programs.
- *license*, a copy of the license to use this material.
- *README*, it is a suggestion to actually read it.
- *righandling*, a directory containing all files for controlling the supported rigs.
- *scopes*, a directory containing the files implementing the various scopes, used by the programs.
- *sound-card*, a directory containing all files for handling sound, based on the portaudio library.
- *utilities*, the directory containing a few, small, support functions, some of them are not even used anymore, but may be of use in future releases or extensions.

- *viterbi*, the directory containing an implementation of the viterbi algorithm, used in e.g. the qpsk and the mfsk decoders.
- tgz files for the w32 and x64 executables together with the required dll's.

3 Building an deploying

3.1 The Windows executables

For windows, two sets of executables are available. The sets are generated for 32 bits windows and 64 bits windows respectively.

The executables are

- swreceiver.exe
- fmreceiver.exe
- spectrumviewer.exe

In the package for the 64 bits version, there is one dll enclosed that is needed for execution of the programs. In the package for the 32 bits version, there are two dll's enlosed that are needed for execution of the programs.

The 32 bit version will run on older 32 bit XP machines. Compile-time parameters were set to minimize resource requirements. However, the FM receiver may show some stuttering in the output.

On a 64 bit machine, the 64 version will outperform the 32 bit version. The main advantage of the 32 bit version is the availability of 32-bit dll's for handling additional hardware.

Zadig drivers Note that for driving the pmsdr kit with the built-in drivers as well as for driving the rtl2832U based dabsticks, a winUSB driver has to be installed. Zadig is a support program for installing such a driver.

For driving the pmsdr kit through the external dll, this driver is not needed.

3.2 Generating executables for Linux

Building an executable for the swreceiver is - unfortunately - not (yet) through the simple "configure/make" sequence with the Gnu autotools. The distribution relies on the qmake tool, a tool in the Qt suite, provided with Qt for generating a makefile, including the required MOC compilation.

The directory *swreceiver* contains the main sources for the sw receiver, the directory *fmreceiver* contains the main source needed for the fm receiver. The directory *spectrum-viewer* contains the main sources for the spectrum viewer.

3.3 Building under Ubuntu/Fedora

In Ubuntu and Fedora one need to have packages installed for

- *qt4*. When also installing qt-4 designer one can easily adapt the GUI, and installing qt4-designer causes all relevant qt4 packages to be installed as well.
- libqwt5-qt4 and the accompanying files in the development package libqwt5-qt4-dev. Notice that *qwt-6* has some changes that prohibit some files to compile, so ensure that qwt-5.2 is installed.
- *libfftw3* and the development package *libfftw3-dev*.
- *libportaudio2* and the development package *portaudio19-dev*.

- *libftdi1* and the development package *libftdi-dev*.
- *libusb-1.0* and the development package *libusb-1.0-dev*.
- *libsndfile1* and the development package *libsndfile1-dev*.
- *libsamplerate0* and the development package *libsamplerate0-dev*.

The good news is that all of these packages are available on the various repositories for Ubuntu and Fedora. A script for Fedora to prepare the packages is:

```
#!/bin/bash
#
echo "Preparing the environment for Fedora"
echo "get administrative privileges "
echo "you must belong to the sudoers and to the audio pmsdrusb group"
echo " "
id $USER
echo " "
echo "audio/pulse/pmsdrusb group check"
sudo more /etc/group | grep "($USER)"
sudo more /etc/group | grep "audio"
sudo more /etc/group | grep "pulse"
sudo more /etc/group | grep "pmsdrusb"
echo " "
echo " check udev rules "
sudo ls /etc/udev/rules.d/
echo "load dongle udev rules "
sudo cp rtl-sdr.rules /etc/udev/rules.d/60-rtl-sdr.rules
echo "reload udev rules ... safe reboot! "
echo " "
echo "install packages"
sudo yum install gcc gcc-c++ \
   qt qt-devel qt qwt qwt-devel \
   fftw fftw-devel \
   alsa-lib alsa-lib-devel alsa-tools portaudio portaudio-devel \
   libsndfile libsndfile-devel libsamplerate libsamplerate-devel alsa-plugins-samplerate \
   libusb1 libusb1-devel libftdi libftdi-devel
#prerequisites install check
echo "
"
uname -a
echo " "
rpm -qa | grep "gcc"
echo "
"
rpm -qa | grep "qt"
echo "
"
rpm -qa | grep "qwt"
echo "
"
rpm -qa | grep "fftw"
```

```
echo "
"
rpm -qa | grep "alsa"
echo "
"
rpm -qa | grep "portaudio"
echo " "
```

The script for ubuntu reads

```
#!/bin/bash
#
echo "Preparing the environment for Ubuntu"
echo "get administrative privileges "
echo "you must belong to the sudoers and to the audio pmsdrusb group"
echo " "
id $USER
echo " "
echo "audio/pulse/pmsdrusb group check"
sudo more /etc/group | grep "($USER)"
sudo more /etc/group | grep "audio"
sudo more /etc/group | grep "pulse"
sudo more /etc/group | grep "pmsdrusb"
echo " "
echo " check udev rules "
sudo ls /etc/udev/rules.d/
echo "load dongle udev rules "
sudo cp rtl-sdr.rules /etc/udev/rules.d/60-rtl-sdr.rules
echo "reload udev rules ... safe reboot! "
echo " "
echo "install packages"
sudo apt-get install gcc g++ \
   libqt4-dev libqwt5-qt4 libqwt5-qt4-dev \
   libfftw3-3 libfftw3-dev \
   alsa-base libasound2 libasound2-dev alsa-utils libasound2-plugins \
   libportaudio2 libportaudio19-dev \
   libsndfile1 libsndfile1-dev \
   libsamplerate0 libsamplerate0-dev \
   libusb-1.0-0 libusb-1.0-0-dev \
   libftdi1
#prerequisites install check
echo "
"
uname -a
echo " "
rpm -qa | grep "gcc"
echo "
"
rpm -qa | grep "qt"
echo "
"
rpm -qa | grep "qwt"
```

```
echo "
"
rpm -qa | grep "fftw"
echo "
"
rpm -qa | grep "alsa"
echo "
"
rpm -qa | grep "portaudio"
echo " "
```

Then, "cd" to the directory of choice

```
.../sdr-j-4.0/make-w32/swreceiver
```

or

```
../sdr-j-4.0/make-x64/swreceiver
```

There are slight differences in the places where Ubuntu resp. Fedora keep some of the include files as well as in the naming of some of the libraries to be included. Each of the ".pro" files contains two specifications, one for ubuntu and one for fedora. Each specification is of the form

```
unix {....}.
```

with a text comment line telling which Linux is meant. Commenting one specification out is by preceding the lines with a sharp-sign (#). Comment/uncomment the section for Ubuntu or Fedora.

For Fedora, one should run

```
qmake-qt4
```

while for Ubuntu one runs

```
qmake
```

Now, the makefile is generated and if all is well a simple

```
make
```

Please note that for actually running the program with a receiver connected to the USB gate, one needs to have sufficient privileges. My personal script is a "sudo ..." one.

3.4 Building for Windows

The Windows exectables are created using the cross compilation facility available under Fedora(-17). The facility supports cross compilation of Qt based programs. The packages required all are easily built using the facility, and, when installed, mingw32-qmake-qt4 or mingw64-qmake-qt4 will create a Makefile. Although quite some work, it works great!!

3.5 Deploying the program

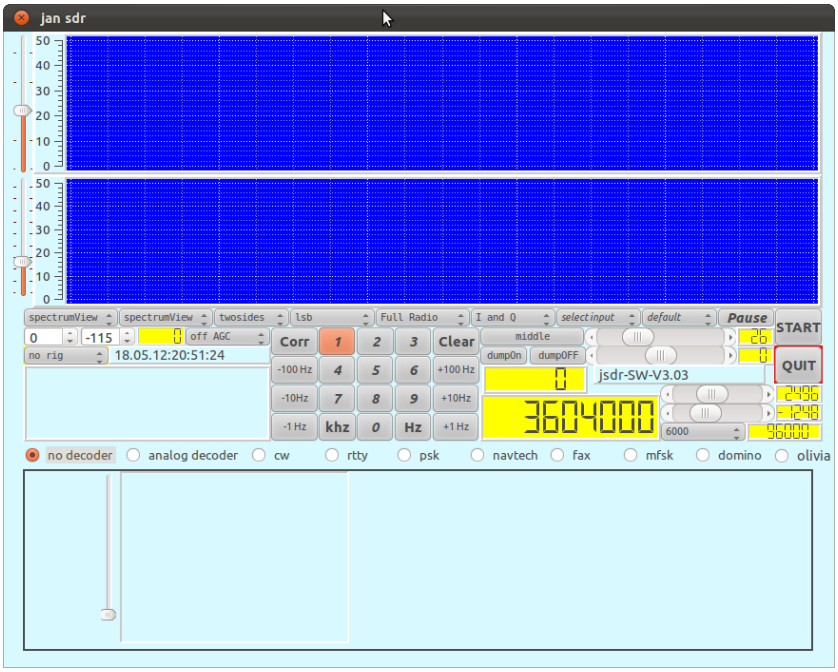
Once the executables are available, starting the program is either by clicking on the icon or by using a command window. For windows deployment, ensure that the required dll's (both the dll's that are part of the distribution as well as dll's that are to be loaded to control specific hardware) are in the searchpath. It is recommended to keep these dll's in the directory where the executable resides.

When running the sw receiver from the command line, one may pass switches to set some configuration values

- *-S* samplerate is 48000, outputrate set to 24000;
- *-R* samplerate set to 96000, outputrate set to 48000 (default);
- *-T* samplerate set to 192000, outputrate set to 48000;
- *-i filename* use *filename* as ini file;
- *-n* do not use an ini file, use the programmed defaults (default);
- *-o filename* at program termination, create an in file *filename*;
- *-d* do not use an ini file and create at program termination a new default ini file with the settings at program termination;

4 Description of the shortwave radio program screen

Once the program is started, the following screen will appear



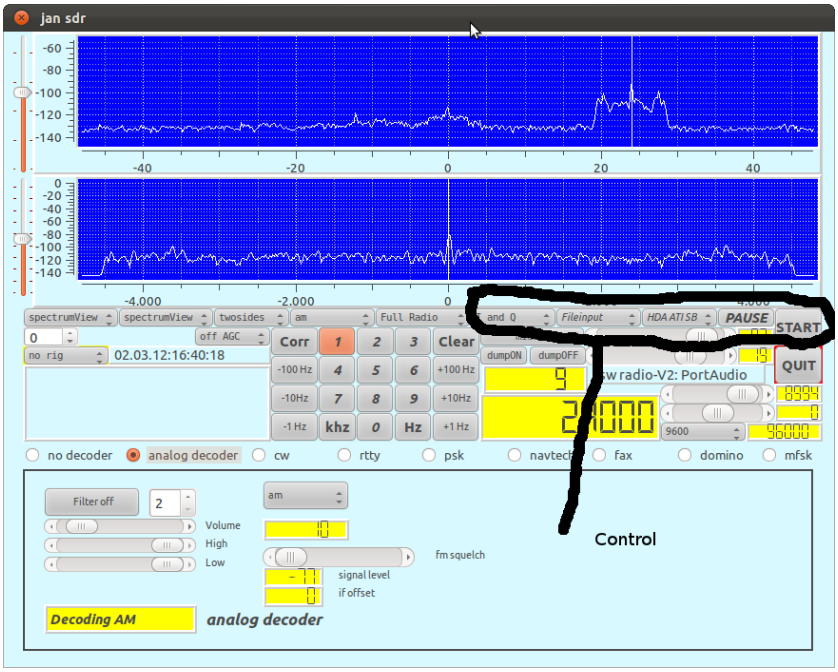
The screen shows three areas:

- *on top*: two displays, one displaying the spectrum for the whole band covered, i.e. with the width of the selected samplerate, the second one displaying the spectrum for the baseband selected for the decoder.
- *in the middle*: buttons and sliders to control the program and the associated radio device.
- *on the bottom*: a screen for the decoder. For each of the different decoders, a separate screen is defined and shown upon selection of the decoder.

5 Control

5.1 Selecting input and output

The program, once started, will set switches and selectors to a default value, or - if specified - according to the readings of an ".ini" file. If input and output streams are set properly - either through the ini file or through manual settings - the start button may be activated by clicking on it. Pushing the start button, however, without having selected appropriate input and output streams results in a message with a request to set the streams properly.



The elements, indicated in the picture above, are

- *selection of the input stream.* Obviously, the possibilities for selection are dictated by the system configuration. Selecting an inputstream is - obviously - not required when selecting a file to be read as input or if a device is selected (e.g. dabstick) that is itself under control of sending data through the usb.
- *selection of the output stream.* Again, the possible selections do depend on the system configuration.
- *start button.*

The rate with which the input stream is processed is indicated on the screen.

When appropriate input- and output streams are selected, clicking on the start button will cause the program to actually start reading samples from the input stream and to process these samples.

Personally, I am using an EMU-202 box (since I blew up the soundcard in my laptop with an experiment with a one-tube receiver), which is selected for the input, while the selected sound output is through the laptop-card.

Linux and Portaudio Linux and soundcard handling - done through Portaudio - are no real friends. The combo-boxes for the input and output streams show a lot of potential streams, most of them do not work properly. Experience shows that selecting the hardware, i.e. the "raw" entries to the cards,

or the default option does work best. Often, when opening the program quite some error messages on either blue tooth or mixers are given, these can be ignored. The PA people are working on it.

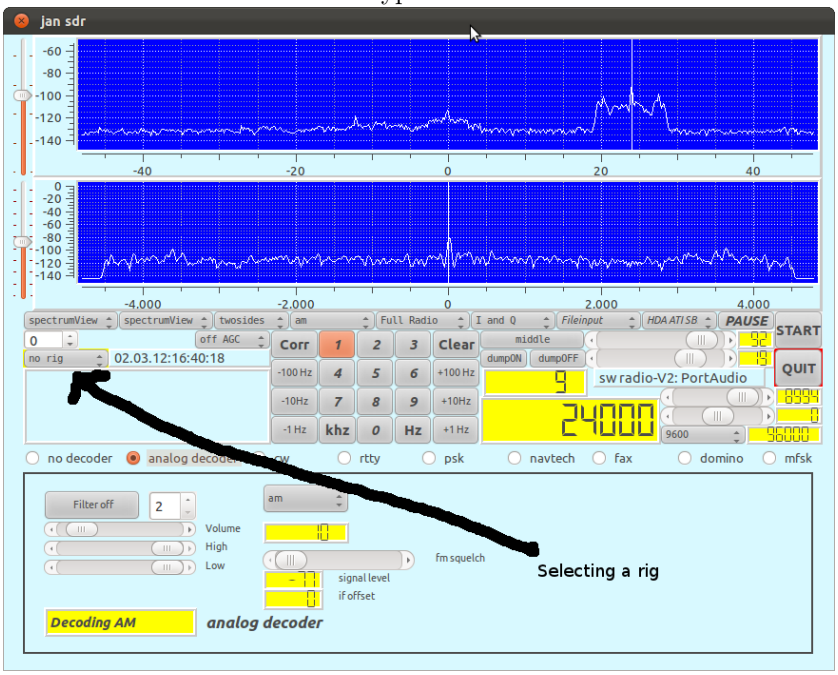
Pulse, the apparent default sound handler in Linux environments and portaudio hate each other even more, do not even try to select *pulse* as either input- or outputstreamer. I have been struggling with Pulse for input and output. However, I did not manage to get reasonable results, Pulse seems to provide data in large chunks, too large to show a more or less continuous stream. Any help to address this is greatly appreciated.

Windows and Portaudio The combo-boxes for the input and output streams show a lot of potential streams. I am using the ASIO software interfacing to the Emu-202 card for input and the card of my laptop for output. (ASIO, when used for both input and output through the EMU requires input and output with the same speed settings). The default configuration for Portaudio used here does not allow Asio for both input and output (code for the implementation for ASIO as both input- and output device is commented out in the software)

Windows/ASIO/portaudio/Emu-202 runs 192 K sampling without problems, while Linux/Alsa/portaudio/Emu-202 is limited to 96K.

5.2 Selecting a radio device

When selecting a radio device, one can choose a *no rig*, (selected by default), *replay*, i.e.file input, the Italian *PMSDR* and rtl2832U based *dab sticks*. When running the windows 32 bits version, an option is to select "extio". This option will allow to connect to various other hardware devices through the use of specific dll's. Dll's that are supported describe hardware type 4, with the exception of the ExtIO_RTLSDR.dll which is of type 3.

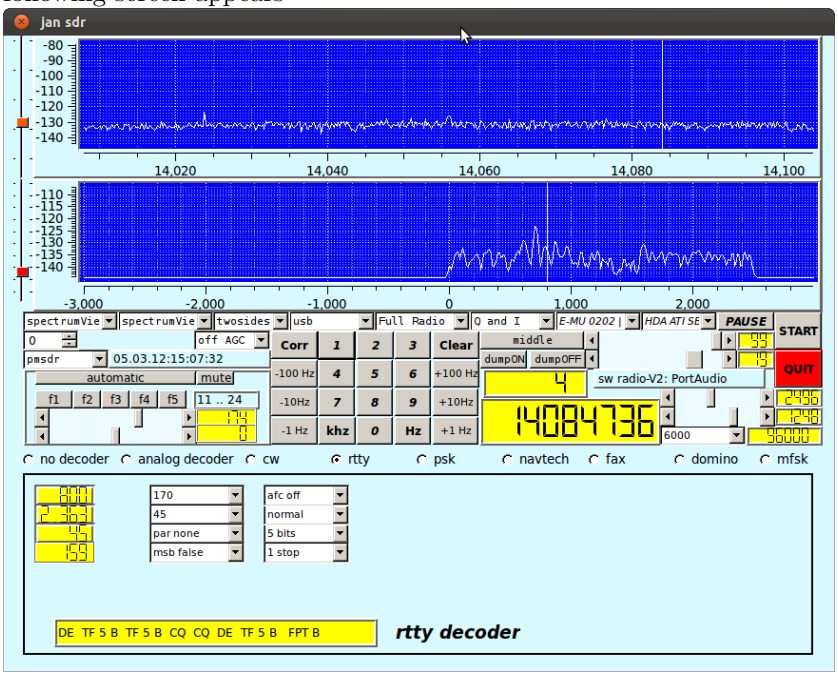


Essential is that when choosing a real device, the selected device is physically connected through the USB. On selection, the program is starting communication with the device and the device cannot be found, or cannot be opened, an error message is given and the "no rig" pseudo device will be selected again.

Note furthermore that the program, when connecting to one of the mentioned radio devices, needs to have sufficient permissions for the usb connection.

Note that for some usb devices one has to install a winUSB driver through the zadig program.

Once the (physical) device (radio) is selected and (successfully) connected, a subscreen will appear with additional control buttons for the selected device. As an example, when selecting the pmsdr, the following screen appears



The subscreen shows some specific controls, such filter selection, muting or not, and a bias slider.

5.3 Tuning and selecting the bandpass filter

Once a radio device is selected and the program is running, data is passing through the system and it is time to select a frequency and to define the passband.

For *selecting* a frequency, the keypad is available. Frequencies can be specified in KHz, in which case the KHz button is to be pushed after entering the frequency, or in Hz, in which case the Hz button is to the one to be clicked upon. A simple check is built in that the selected frequency is supported by the selected rig. A VFO frequency is sent to the radio device, affecting the settings of the radio.

One might notice than when setting the frequency through the keypad, a needle in the top display will be positioned at 3/4 of the screen, indicating the selected frequency. Indeed, the position chosen reflects 1/2 of the right half of the screen. Also note that the middle of the screen corresponds to the selected VFO frequency.

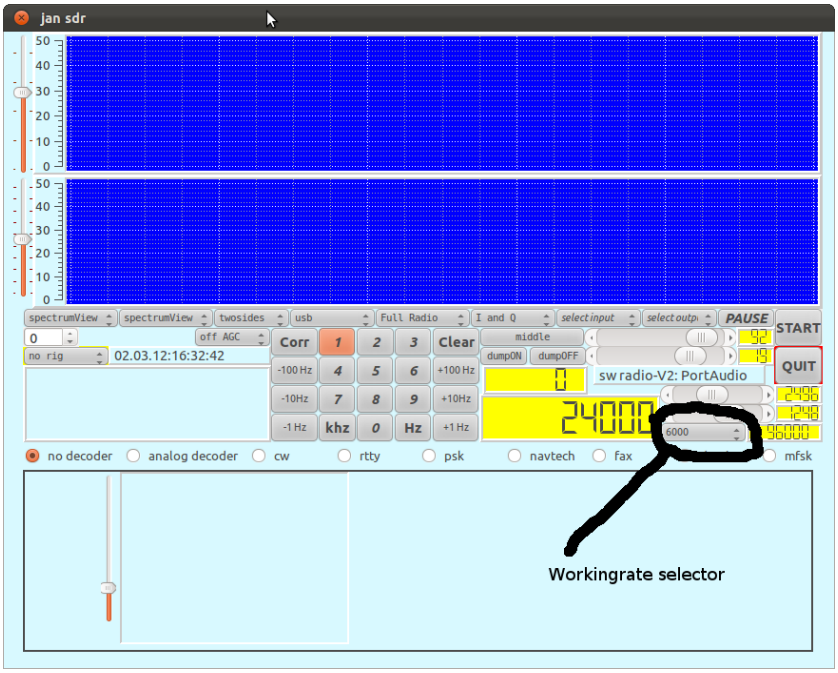
Changing the frequency can be done in different ways:

- a new frequency can be keyed in;
- the +100, +10, +1, -1, -10, and -100 buttons can be used to shift frequency with these amounts;
- a mouse click on one of the scopes will change the frequency as well.

One might also notice that when changing the frequency a small amount, the needle on the screen will move to another place, while the VFO frequency remains unaltered. Finally, one might notice that when a sequence of changes would cause the needle to shift off the screen, the needle is reset to the 3/4 position, while the VFO frequency and thus the frequency indications along the X-axis, change.

The next thing to consider is the *width of the passband* and the *position of the passband* within the given spectrum. What must be taken into account is that the samplerate with which samples are passed to the decoding modules provides a natural upperbound for the bandwidth of the passband.

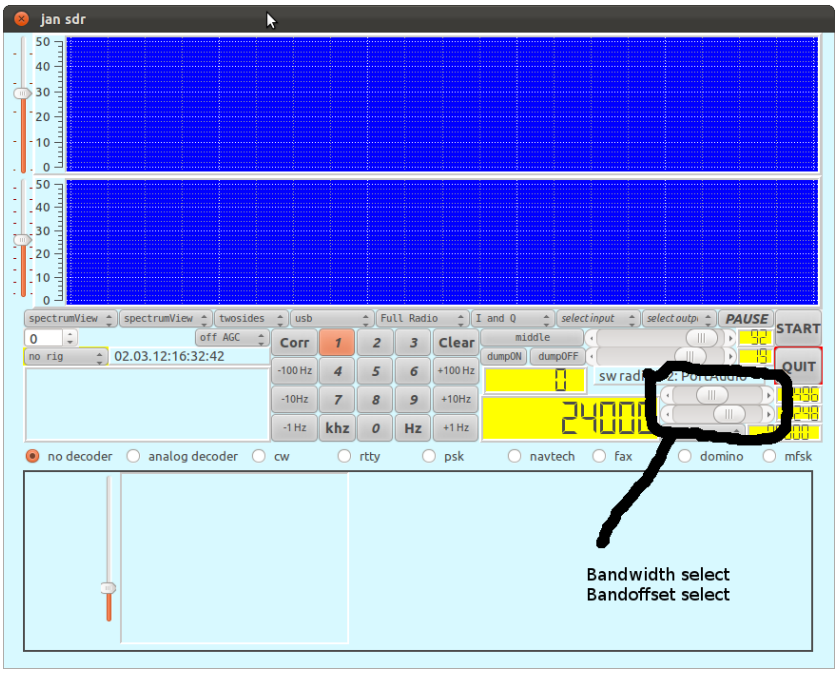
One might change the natural upperlimit of the passband by setting the *workingrate* to a different value. The workingrate selector predefines a number of workingrates, i.e. 4000, 6000, 8000, 9600, 12000, the default is 6000 since that works fine (for me) for the modes used on shortwave.



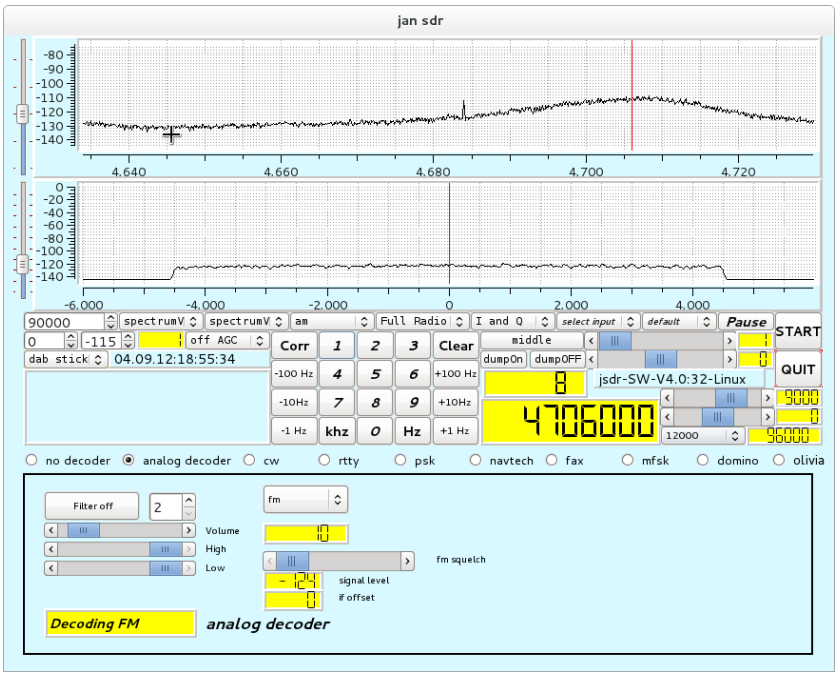
Within this band, the actually selected sub band is defined by the *bandwidth slider* and the *bandwidth offset*. One of a number of predefined bandwidth settings can be chosen by the *bandwidth selector*, e.g. usb for a band of 0 .. 2500 Hz, and a band of 1000 Hz for fax reception. The default setting for this combo-box is USB.

The center (i.e. the precise middle) of the selected band can be modified by the *bandwidth offset slider*, changing this offset results in shifting the selected band.

The width of the selected band can be modified by the *bandwidth slider*.

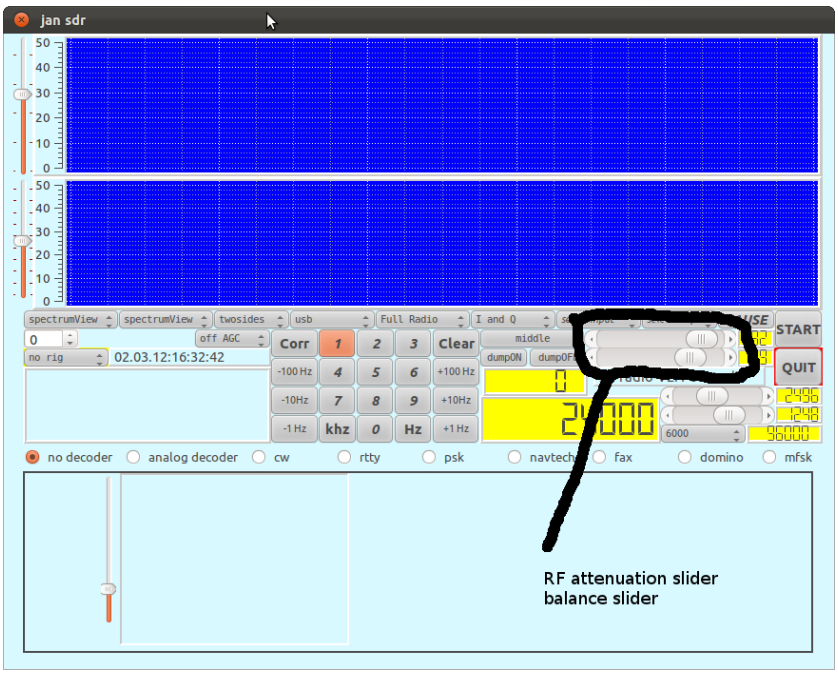


For selected devices (currently the dabstick only), there is the possibility of selecting an *offset*. Consider an upconverter that converts the frequencyband from 0 .. 30 MHz to 116 to 146 Mhz. Setting the offset to 116000 KHz allows the user to choose frequencies in the range 0 .. 30 MHz, while the actually tuned frequency is 116 MHz higher.



5.4 Attenuation

Software attenuation of the incoming signal is through the *attenuation slider*.



The relative attenuation of the left and right input channel can be influenced by setting the *balance slider*, moving it to the left means increasing the strength of the left channel, at the same time decreasing the strength of the the right channel, and vice versa. The bottom slider in this row as indicated in the figure, is removed in the current version.

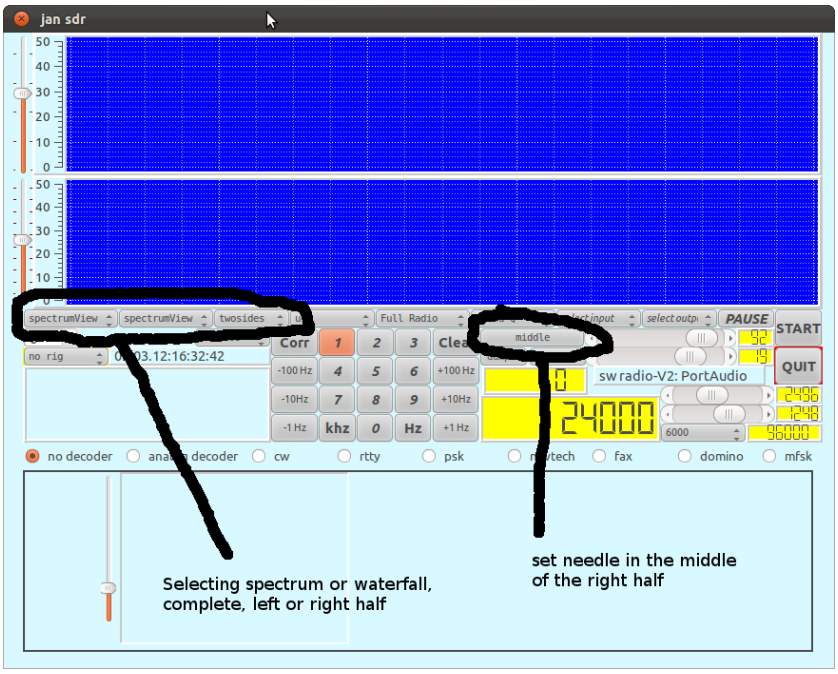
On the emu the left and right channel are controlled independently by a volume knob. These controls are too coarse, and I use the balance slider for a precise tuning the strength of the I and Q signals such that mirror rejection is maximal.

5.5 Viewing spectra

By default, the two displays show the spectra of the data. Setting the *scale* of the amplitude of the spectra can be done by the sliders on the left side of the displays. Data is input with an accuracy of 24 bits, the scale is therefore app 140 db.

Choosing a *waterfall* view for a spectrum rather than a two dimensional view is by selecting the *waterfall* option on the appropriate combo-boxes.

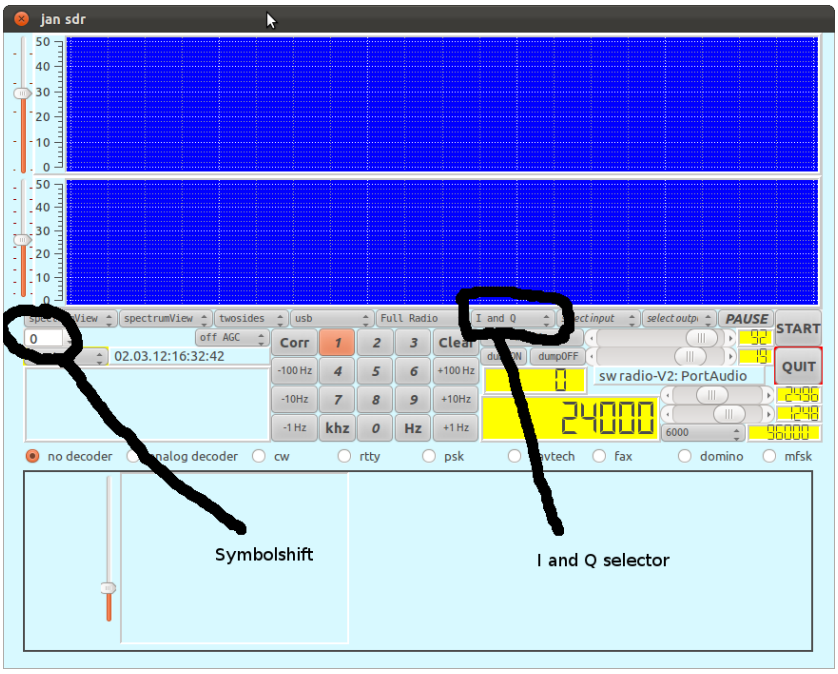
For the bottom one of the two displays, i.e. the display showing the spectral data of the data sent to the decoder modules, there is an additional option, one may select either the full spectrum, the positive half or the negative half.



One additional button is of interest when looking at spectra. The button labeled *middle* will ensure that the selected frequency is displayed in the middle of the right half (the positive side) of the spectrum display and the VFO adapted accordingly.

5.6 Setting I and Q

Obviously, some default is chosen when mapping the input channels onto the I and Q signals from the radio device. The combo-box indicated in the picture below, allows one to select either the I or Q signal as built-in, or an interchange between the I and Q channels. The spinbox indicated in the picture provides the possibility of shifting the I and Q symbols relative to each other, i.e. setting the spinbox to 1 causes the I symbol sequence to be delayed by one.

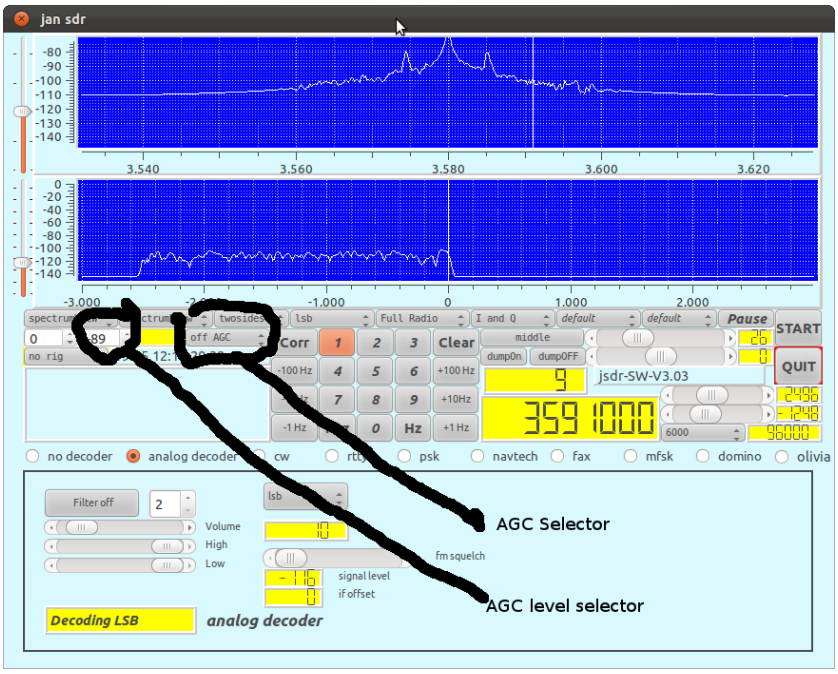


Two remarks:

- Different radio devices might have different views on how to map I and Q to left and right. The I and Q for the Elektor SDR card are the opposite from the I and Q from the PMSDR.
- The FM receiver switches the I and Q automatically when encountering frequencies above 55 Mhz (i.e when using the 3-d harmonic signal as VFO).

5.7 Setting AGC

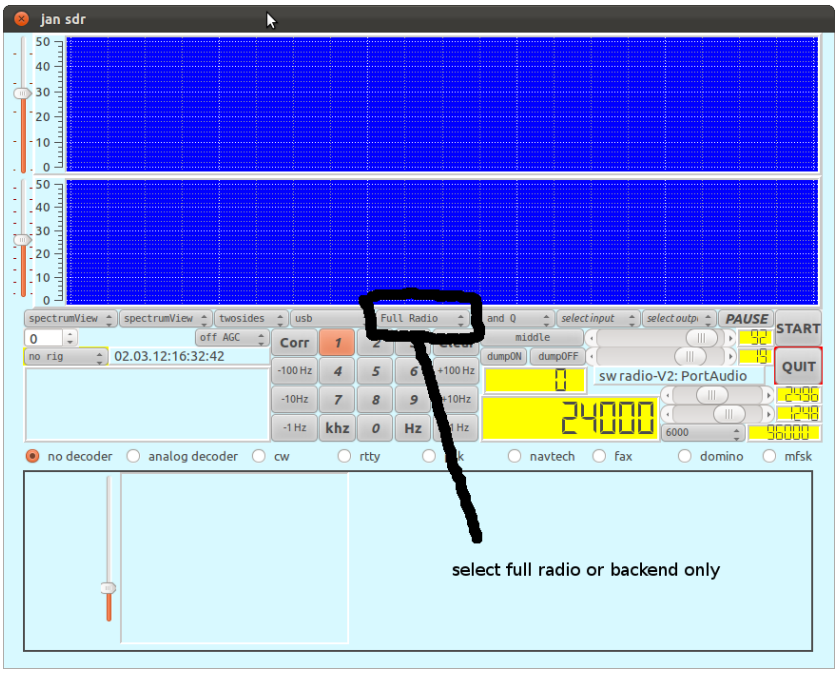
For AM reception, AGC might be useful. In the current implementation, AGC gain is reduced when the signal strength is above a certain, specified, level. The level at which AGC will be activated - obviously when switched on - is set by the AGC level spinbox. Selecting the AGC (either *AGC off*, *AGC slow* or *AGC fast*) is by the appropriate combo-box.



Left of the AGC selector, one finds a setting indicating the threshold of the AGC.

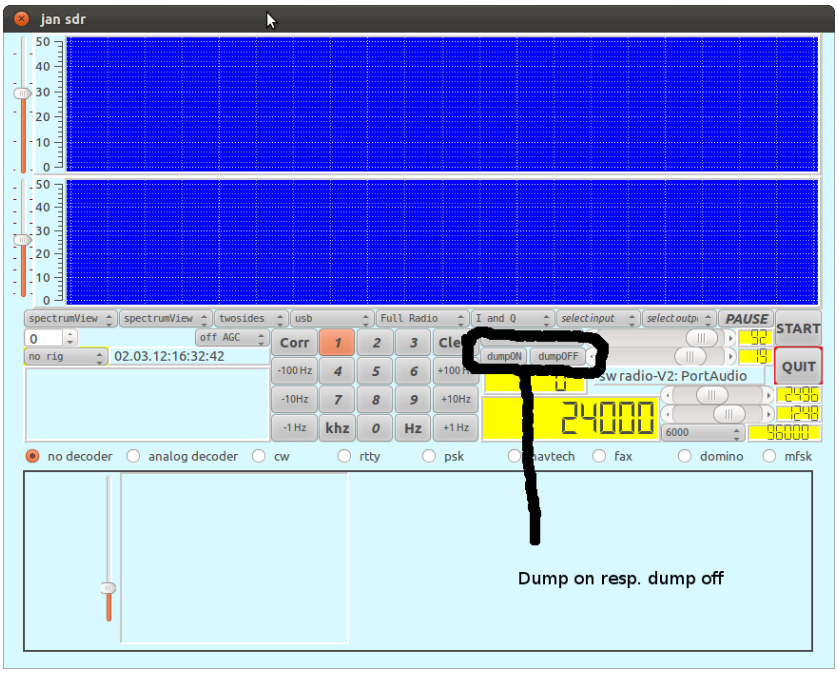
5.8 Setting full Radio

Normal operation of the software is as full radio. An option is to select the *back end* only. When choosing this option, the front end functions are disabled, and the input stream is - after decimation - passed on to the decoding back end. This is useful when using the software in combination with a regular shortwave receiver or, as exemplified in the picture below, when exercising decoder functions using widely available sound files as input.



5.9 Dumping the inputstream

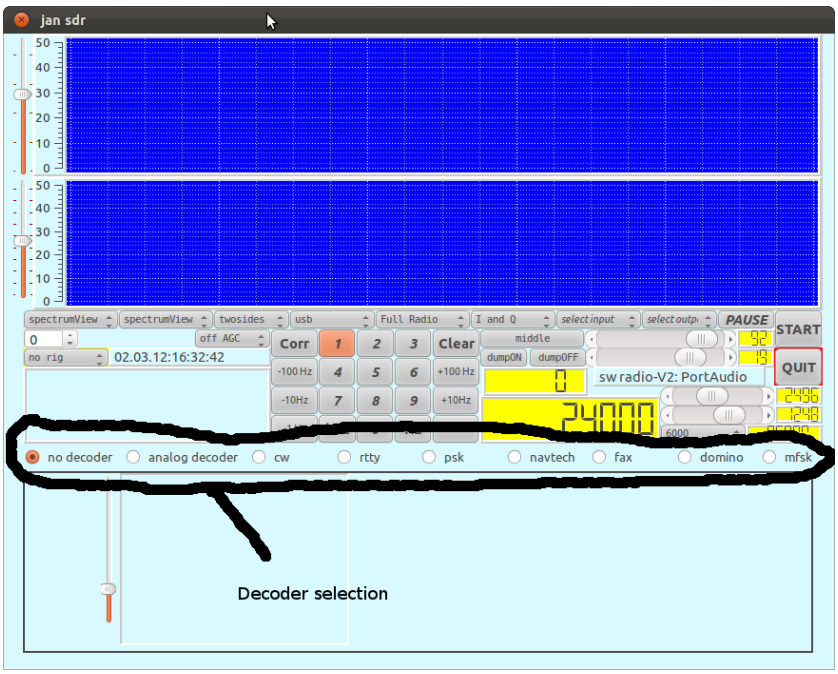
In order to allow repetition of experiments, a simple mechanism for dumping the input stream is implemented. Pushing the *dumpON* button will invoke a file selector and once a file(name) is chosen (or given), the input stream will be written - unaltered - to this file. The format with which the data is written is the regular ".wav" format, the baudrate is the baudrate the receiver is operating upon. Pushing the *dumpOFF* button causes the file to close and turn the mechanism off. Note that in the current version there is no indication when the data is dumped.



The resulting ".wav" file can be used as input through the "replay" mechanism of the receiver.

5.10 Selecting a decoder

Front end processing transforms the signal into a baseband signal with a bandwidth as specified in the workingrate selector (the selector to the left of the value display showing the samplerate). The signal is filtered to the bandwidth specified and can be decoded.



The default decoder is the "no-decoder", this (non-)decoder will just transform the incoming data up

to the samplerate needed for the output card, while displaying the decoder input in two dimensions (i.e. the I and the Q) on a small screen.

A decoder can be selected by clicking the requested decoder (radio) button.

6 Handling decoders

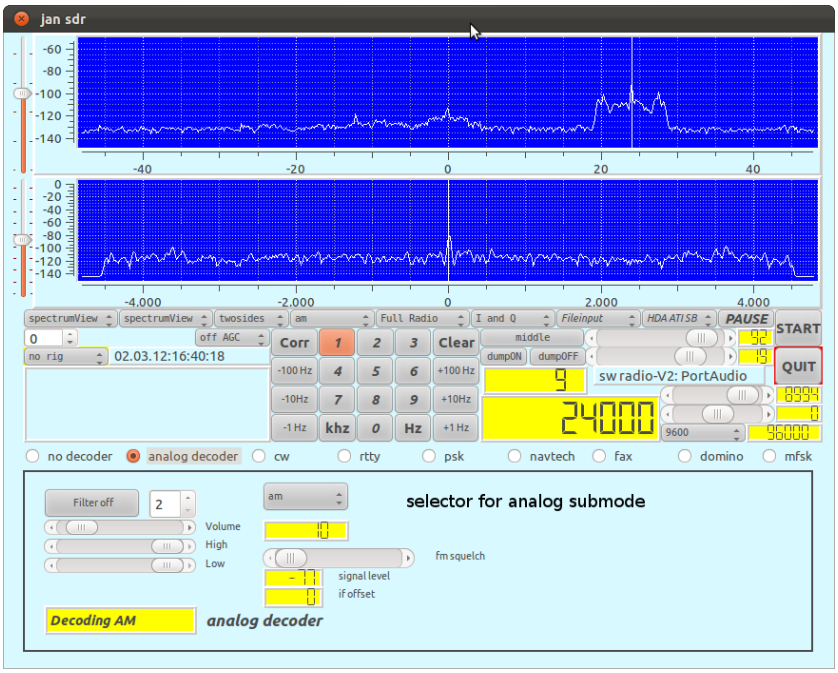
6.1 Analog decoder

In the mode "analog decoder", the following analog decoding modes are available:

- *usb*;
- *am*;
- *lsb*;
- *smallband fm*;
- *isb*, double side band detection;
- *synchronous am decoder*.

Notice that selecting an analog decoding mode such as *lsb*, does not change the setting of the passband. Filtering is such that when choosing e.g. the *lsb* decoding mode while having selected the *usb* passband, hardly any signal will be heard.

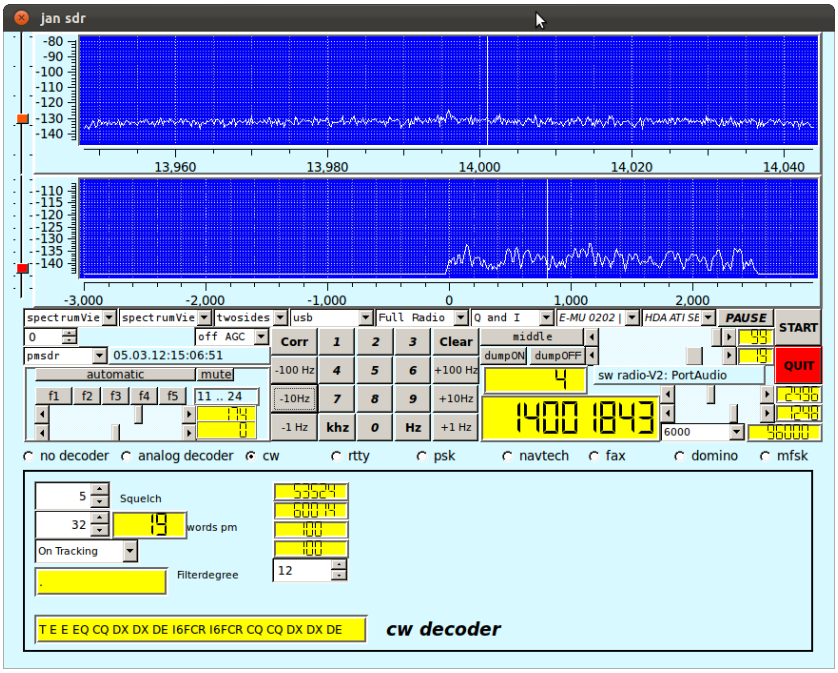
The sliders speak for themselves. It must be noted that the squelch is automatically invoked in the fm decoding mode, but squelch sensitivity can be set by the slider.



6.2 Cw decoder

The cw decoder decoder will attempt to guess the speed, however, up to a limit specified by the user. Setting an upper limit to this speed is with the selector left to the "words per minute" display. Sometimes it is beneficial to increase this value, its default value is 32.

Notice that the working IF for this decoder is chosen to be 800 Hz, tuning to a station that sends on frequency X then requires setting the receiving frequency on X - 800.



The *squelch* is used to set a threshold in the signal, below which it is considered noise.

The *filterdegree* can be set, it is default set to 12, which is a reasonable value for the type of filter (Butterworth).

The four boxes displaying values:

- the top two indicated the estimated length of the dot and the space.
- the bottom two indicate (clamped) levels for the peak of the signal and the estimated noise level. These value displays are merely for debugging purposes.

6.3 Rtty decoder

The working IF for the Rtty decoder is chosen to be 800 Hz, tuning to a station that sends on frequency X implies setting the receiving frequency on X - 800.

Tuning should be such that the 800 Hz needle should be precisely in the middle between the two legs of the spectrum of the received rtty signal.

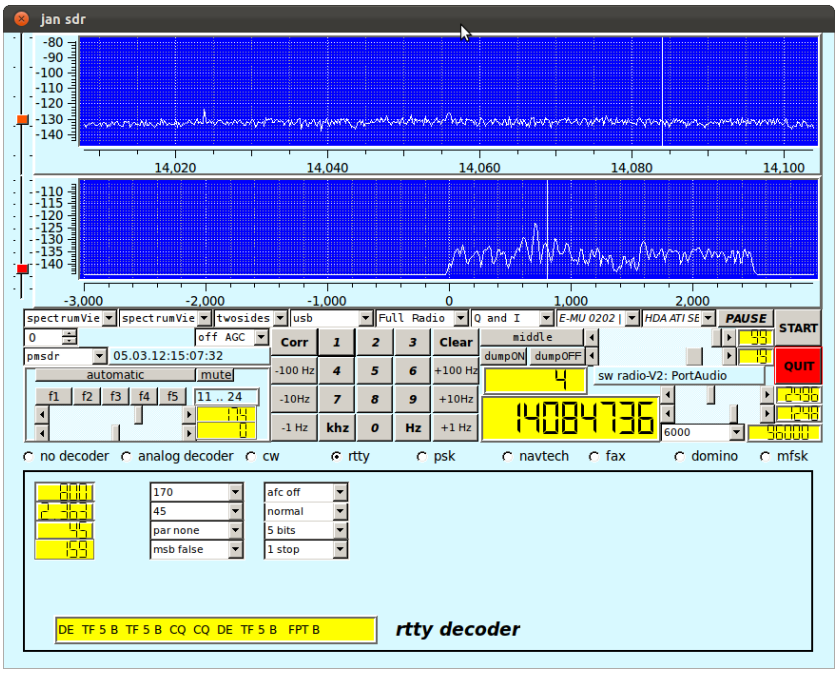
The decoder provides 8 combo-boxes for setting some decoder specific parameters. In two columns, from top to bottom, from left to right:

- the shift, defaulting to 170 Hz. One from a number of alternatives can be selected;
- the *baudrate*, defaulting to 45 Hz;
- the *parity bits*, defaulting to 0;

- the *MSB*, i.e. Most Significant Bit first or last;
- the *afc*, defaulting to off;
- the interpretation mode: normal or reverse, defaulting to normal;
- the *wordlength in bits*, defaulting for rtty to 5;
- the *number of stopbits*, defaulting to 1.

The four value displays indicate:

- the *actual IF*, when the afc is on one might look at the changes in this value and use that as an aid in tuning;
- a very rough estimate of the *frequency error*, not a very useful figure, is being used to test some algorithms;
- the *estimated baudrate*;
- the *estimated shift*.



6.4 Psk decoder

The IF for the psk decoder is chosen to be 800 Hz. This means that for selecting a station at frequency X, the receiver should be tuned to X - 800.

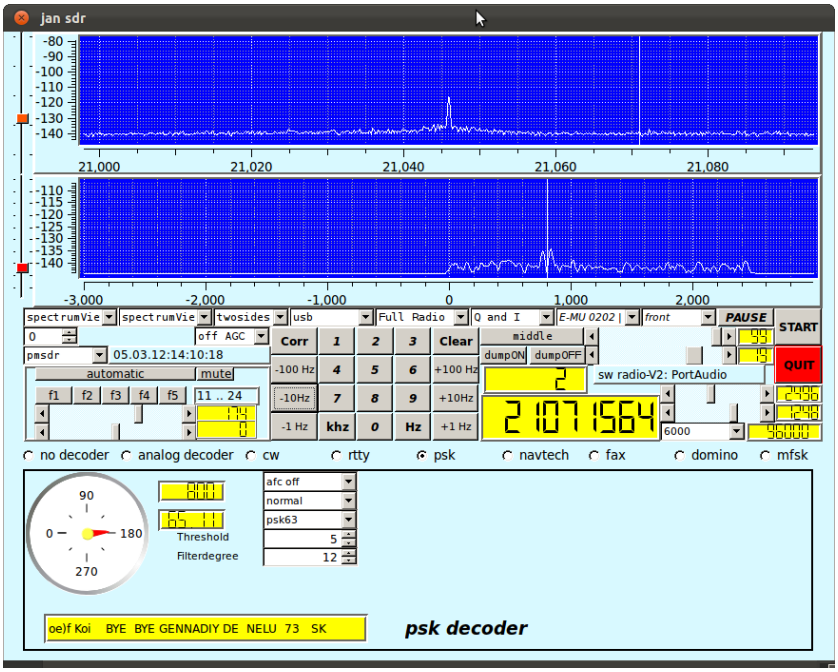
Characteristic for the psk decoder is the dial display at the left size of the decoder specific subscreen. While operating, this display will show the phase angle between - perceived - subsequent psk bits. Ideally the values shown are 0 and 180 degrees.

The parameters to be controlled are:

- the *afc*, allowing the software to change the IF somewhat in order to improve reception;
- *normal or reverse* interpretation of the incoming signal;

- the psk or bpsk mode;
- the *threshold* used to distinguish noise from signal. The scale used has no specific physical meaning.
- the *strength* of the filter used to separate the signal on the IF from other signals in the environment. The degree indicates the order of the Butterworth filter being used.

The two value displays indicate the actual IF and the signal quality respectively.



6.5 Fax decoder

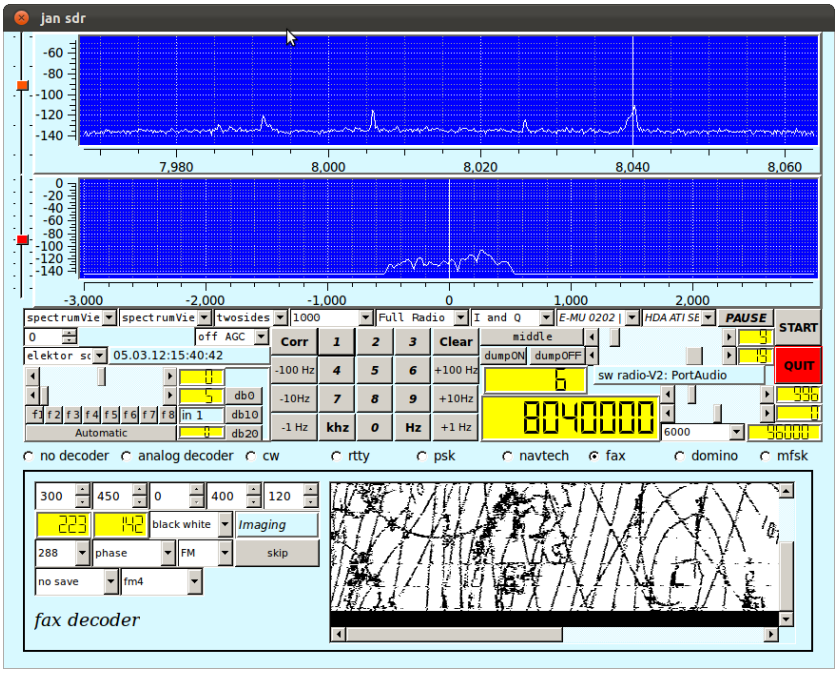
For the fax decoder we have chosen an IF to be 0 Hz, the selected frequency is therefore the frequency in the middle of the fax signal. Fax signals are well bounded by a 1000 Hz filter, the presets for the bandfilter therecore contain a 1000 Hz specification. The fax decoder has a lot of parameters that can be set:

- the top row contains spin-boxes for
  - the *start frequency* for a fax recording, default 300 Hz;
  - the *stop frequency* for a fax recording, default 450 Hz;
  - the *IF*, default 0;
  - the *deviation* in Hz, default 400 Hz;
  - the *LPM* (lines per Minute), default 120.
- the second row has two value displays, a combo-box and a textlabel:
  - the *Linenumber* of the current fax;
  - the current *APT* frequency;
  - the selector for Black and White, Gray or Colour;

  - the label stating the current step selection;

- the third row contains 4 combo-boxes:
  - the *IOC* (Index of Cooperation) Selector, default 288 for wheatherfaxes;
  - the *phase selector*, selecting whether or not the phase is inverted;
  - the *modulation mode selector*, either FM or Am, defaulting to FM;
  - the *skip* button, allowing the current processing step to move on to the next one.

- the fourth row contains two combo-boxes:
  - the *save* button, defaulting to *no save*. If *save* is selected, a filename will be asked for and the fax will be saved upon reaching the end;
  - the *demodulator* selector. Three different demodulators are implemented (just to experiment), this combo-box alles selecting one.



6.6 Mfsk decoder

As with the other decoders, the IF for the domino decoder is chosen to be 800Hz. This means that for selecting a station at frequency X, the receiver should be tuned to X -800.

Tuning to an mfsk transmission is quite hard, therefore the decoder screen contains two additional displays that may help. The top display, indicated as *Tuning indicator* contains three elements, two red fields with a green field in the middle. The red field at the left indicates that the maximum of the presumed signal is too low a frequency to fit, the red field at the right the same for too high a frequency. In the picture below, the tuning indicator indicates correct tuning.

Below this tuning indicator, a small waterfall display shows the tones of signal in more detail. It shows 10 more tones than the mfsk signal is supposed to have such that it might help in correctly tuning the signal.

The decoder provides 3 combo-boxes for setting parameter values:

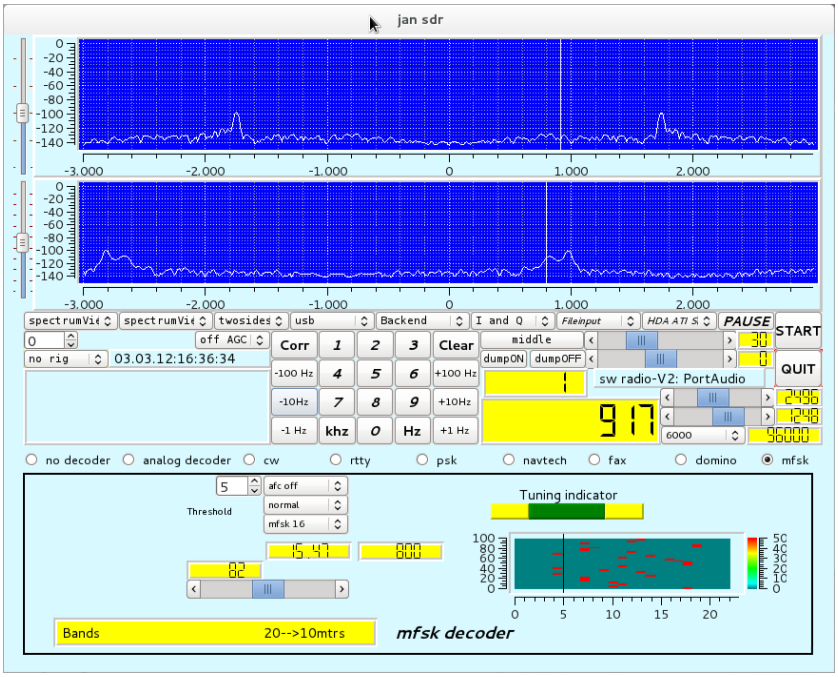
- the afc, by default off;
- whether the incoming signal is to be interpreted normal or reversed w.r.t the base tone;
- the mfsk mode.

Furthermore, the decoder has a box with which a threshold value can be set, below which incoming signal is to be considered noise.

The slider on the screen can be used to set the IF, for normal operation it is hardly useful. It is reminiscent of some previous experiments.

The three value displays on the screen show:

- the - perceived - quality of the signal;
- the estimated baudrate;
- the IF, which only changes when the afc is set.



Notice that, since in the period of writing no real mfsk transmission was caught, the screen for this mode was made while processing a file with the backend option of the software.

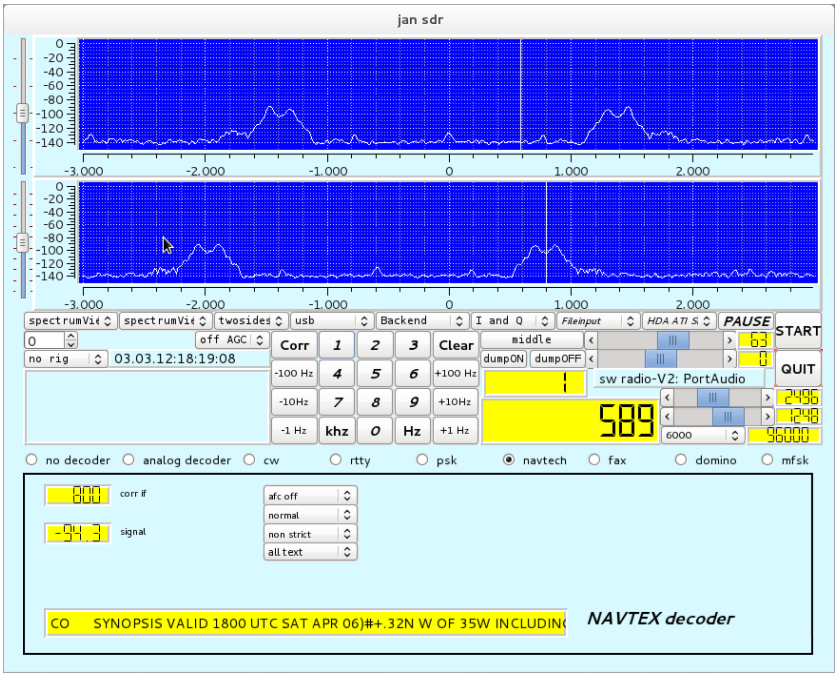
6.7 Navtex decoder

Navtex decoding uses an IF of 800 Hz, so the tuned frequency should be 800 Hz less than the sending frequency. The parameters, settable though combo-boxes, are:

- afc, default off;
- normal or reverse interpretation of the incoming signal;
- strict FEC or non-strict FEC.
- message or general text. In case "message" is chosen, the messages displayed are the proper navtech messages enclosed between "ZCZC" and "NNNN" sequences.

The value displays indicate:

- the actual IF (will only change obviously when afc is set);
- the *strength* of the signal.

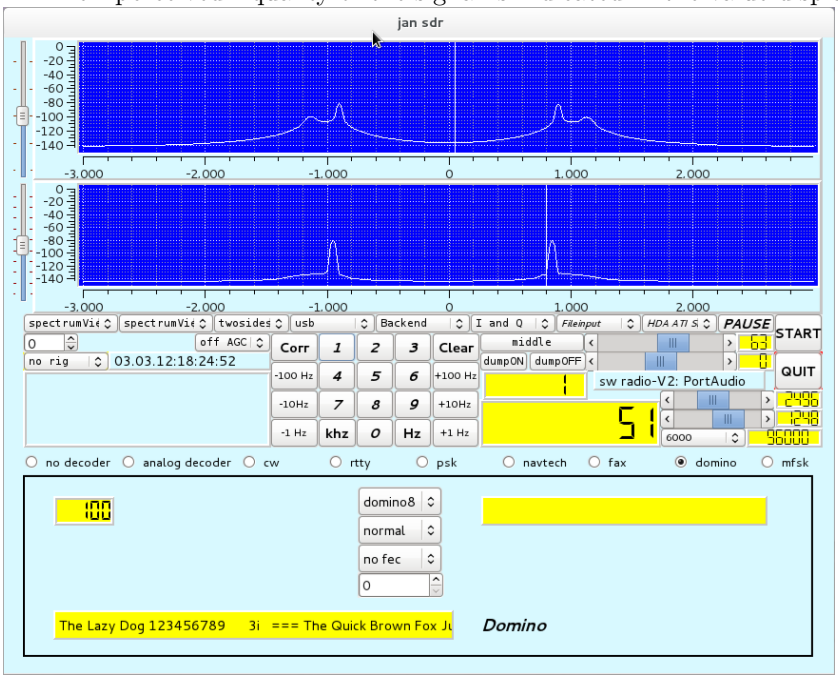


6.8 Domino decoder

As with the other decoders, the IF for the domino decoder is chosen to be 800Hz. This means that for selecting a station at frequency X, the radio should be tuned to X -800.
Domino is by itself an easy mode, tuning is - within limits - not very critical. The various combo-boxes (from top to bottom):

- selecting the domino mode and speed;
- normal or reverse interpretation;
- with or without FEC;
- setting a squelch-level for the decoding.

The - perceived - quality of the signal is indicated in the value display.



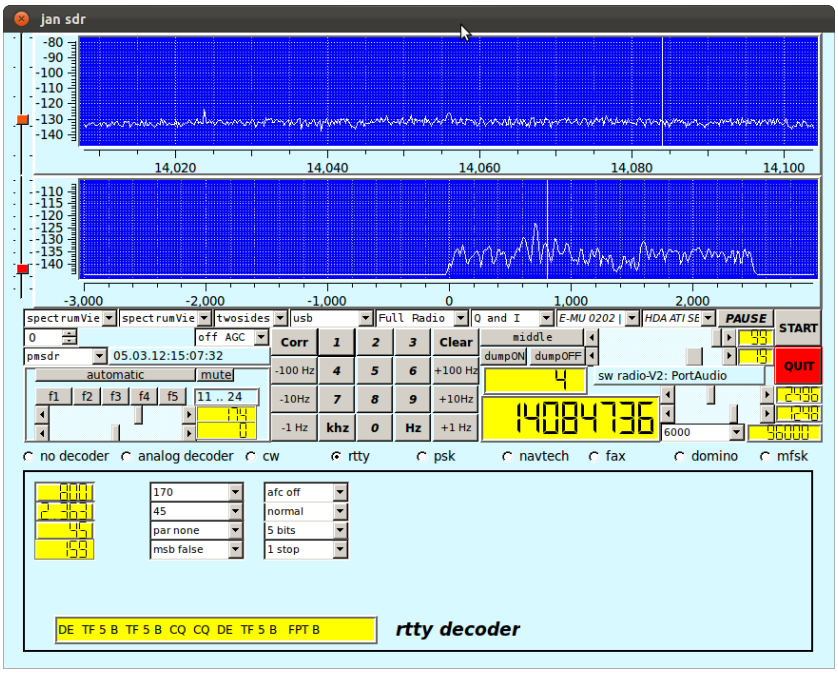
Notice that, since in the period of writing no real domino transmission was caught, the screen for this mode was made while processing a file using the backend option of the software.

6.9 Olivia decoder

7 Controlling the radio devices

When the pmsdr is selected as radio device through the rig-select options, a few pmsdr specific controls are shown. The PMSDR-specific controls are

- Mute selector. The hardware supports muting, selectable by the mute button.
- The filter selection. Filters can be selected automatically or manually. In case of a manual selection, the filter *f1* .. *f4* select predefined filters, *f5* selects *no filter*.
- The bottom slider is the bias slider, setting the bias for the PMSDR kit;
- the next slider is the gain slider, which is - unfortunately - not implemented in the recent versions of the firmware of the PMSDR.



Of course, when using some specific hardware through external dll's, these dll's might provide their own interfaces. I.e. selecting the pmsdr through a dll will initiate a separate control window, controlled by the dll.

8 ini files

The initial settings of most of the sliders and buttons can be specified in the so called ini files. Such an ini file is read and processed in the initalization phase. If no ini file is specified, each of the sliders and buttons will be set to some default value.

When using the -d option as commandline parameter, the settings of the relevant sliders and buttons will be stored in the default ini file, which can be found at

`$(HOME)/.jsdr-sw.ini.`

Starting the program with a different ini file is made possible through the -i switch (-i ¡filename¿). Writing at program termination the ini file to a user specified file will be effectuated when specifying the "-o filename switch.

The structure of the ini file is straightforward and an example is given below

```
[General]
balanceSlider=0
attenuationSlider=26
HFplotterView=spectrumView
LFplotterView=spectrumView
operationSelect=Full Radio
inputModeSelect=I and Q
AGC_select=off AGC
quickbandSelect=lsb
LFDisplayIndicator=twosides
LFspectrumamplitudeSlider=30
```

```
HFspectrumamplitudeSlider=50
iqslider=0
workingrateSelect=6000
symbol_shifter=0
agcThresholdSlider=-115
streamInSelector="E-MU 0202 | USB: USB Audio (hw:2,0)"
streamOutSelector=default
rigSelect=pmsdr
tunedFrequency=3604
analogDecoderSelect=lsb
analogDecoderVolume=10
analogDecoderLowPass=99
analogDecoderHighPass=98
cwwpm=32
cwTracker=On Tracking
cwFilterDegree=12
cwSquelchLevel=5
pskAfcon=afc off
pskReverse=normal
pskSquelchLevel=5
qpskSelector=psk31
pskFilterDegree=12
rttyBaudrateSelect=45
rttyWidthSelect=170
rttyParity=par none
rttyMsb=msb false
rttyNbits=5 bits
rttyStopbits=1 stop
rttyReverse=normal
rttyAfcon=afc off
amtorAfcon=afc off
amtorReverse=normal
amtorFecError=non strict
amtorMessage=all text
dominoSquelch=0
dominoFec=no fec
dominoReverse=normal
faxIOCSelector=288
faxModeSelector=FM
faxDeviationSelector=400
LPMBox=120
phaseSelector=phase
faxColorSelector=black white
faxstartFrequency=300
faxstopFrequency=450
faxCarrier=0
faxdecoderDelector=fm1
mfskMode=mfsk 16
mfskReverse=normal
mfskAfcon=afc off
mfskSquelch=5
```

```
mfskIFadjuster=0
oliviaTones=32
oliviaBW=1000
oliviaSyncMargin=8
oliviaSyncIntegLen=4
oliviaSquelch=squelch off
oliviaSquelchvalue=5
oliviaReverse=normal
nodetectorButton=0
analogDecoderButton=1
rttyButton=0
cwButton=0
pskButton=0
mfskButton=0
amtorButton=0
faxButton=0
dominoButton=0
oliviaButton=0
```

9 Sources of inspiration

The software uses many ideas - and in quite 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.
- *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
  - 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 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 FM software.
  - The pmsdr interface software (Martin Pernter, Andrea Montefusco), and the Si570 data sheets,
  - 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.

10 Availability and licensing

This software is available under the GPL. It was written by me, updated by me, using many ideas - and even lines of code - from others, as far as I can trace all available under the GPL.

```
/*
 *
 *    Copyright (C)  2010, 2011, 2012
 *    Jan van Katwijk (J.vanKatwijk@gmail.com)
 *    JFF Consultancy
 *
 *    This file is part of the JSDR (ESDR).
 *    Many of the ideas as implemented in this software are derived from
 *    other work, made available through the GNU general Public License.
 *    All copyrights of the original authors are recognized.
 *
 *    JSDR is free software; you can redistribute it and/or modify
 *    it under the terms of the GNU General Public License as published by
 *    the Free Software Foundation; either version 2 of the License, or
 *    (at your option) any later version.
 *
 *    JSDR is distributed in the hope that it will be useful,
 *    but WITHOUT ANY WARRANTY; without even the implied warranty of
 *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *    GNU General Public License for more details.
 *
 *    You should have received a copy of the GNU General Public License
 *    along with ESDR; if not, write to the Free Software
 *    Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 */
```