# RXAMADRM - A LINUX PROGRAM FOR DIGITAL SSTV

Ties Bos - PA0MBO

February 12th, 2014

**Abstract**

A linux program was developed for the reception of digital SSTV picture transmissions conforming to the HAM DRM standard as developed by HB9TLK and used in the Windows programs WINDRM, DIGTRX and EASYPSL. It is modeled after the MATLAB program DIORAMA for the reception of commercial DRM broadcast stations.

Keywords: Digital SSTV - Hamradio - DRM - Linux - Reed/Solomon

## 1 INTRODUCTION

The program RXAMADRM can be used to receive (picture) files in the HAMDREAM standard as developed by HB9TLK [1]. In contrast to the programs based on HB9TLK's code like WINDRM [2], EASYPAL [3] and DIGTRX [4], RXAMADRM runs under Linux and its source code is available. It is published under the GNU General Public License (GPL), so you can freely modify and improve it.

The code was not built from scratch, but is based on the program DIORAMA that was written by Andreas Dittrich and Torsten Schorr [5] from the university of Kaiserslautern. DIORAMA is written for MATLAB, a high level interpreter for complex mathematical operations with very nice graphical and GUI-tools.

Initially DIORAMA was adapted to the amateur standard. However for amateur use MATLAB is rather expensive and to circumvent the use of MATLAB, the MATLAB-specific code was translated to standard C. The choice of DIORAMA as a starting point instead of DREAM (on which all existing Ham DRM SSTV programs are

founded) was based on the paper "*Digital Radio Mondial (DRM) Receiver using MATLAB*" by T. Schorr [6] et.al. Moreover DIORAMA turned out to give a somewhat better performance in the reception of the BBC station on 1296 kHz when compared to DREAM under marginal conditions.

In the design of RXAMADRM the digital signal processing (DSP) code is strictly separated from the Graphical User Interface (GUI) code. The DSP part is a standalone program written in standard C (**drmtst**). The GUI part is a separate program written in the script language Tk/Tcl (**rxamadrm.tcl**). Communication between the programs is one way via standard input/output, i.e. from **drmtst** to **rxamadrm.tcl**. This simplifies debugging and allows an easy way to develop much more sophisticated Graphical User Interfaces in languages/systems other than Tk/Tcl.

For compatibility with the popular DRM/SSTV program EasyPal Reed/Solomon decoding was added, using separate programs for the various degrees of encoding. As of version 1_5 RXAMADRM generates a socalled bsr.bin file when a picture file is not received completely. This file conforms to the format used in EasyPal.

# 2 DESCRIPTION OF THE PROGRAMS

## 2.1 The DSP program drmtst

The signal processing tasks that must be performed are:

1. acquisition of 400 msec worth of sound data
2. presentation of the signal in the audio passband of the receiver in a "waterfall" picture
3. demodulation/equalization
4. channel decoding
5. source decoding

These signal processing tasks are gathered in one standalone program called **drmtst**. Its main routine has an infinite loop in which the subroutines that perform these 5 tasks are executed in sequence repeatedly.

### 2.1.1 acquisition of sound data

The program **drmtst** is a real-time program, it calculates its result (the image file that was sent) **during** the reception of the signal. As the audio signal comes from the receiver continously, but has to be

analyzed in chunks, some way has to be found to read the sound data into the computer and analyze it simultaneously.

In most Linux systems the recording of sound input via a soundcard is handled by the ALSA soundsystem. This system sees to it that the sound signal which is presented to the microphone or line input connector of the soundcard is sampled by an Analog to Digital Converter (ADC) and stored in an internal buffer. Application programs like **drmtst** have to see to it that the content of this internal buffer is processed before it overflows and part of the incoming signal is lost. Unfortunately the size of the internal buffer of ALSA is much smaller than corresponds to the duration of the sound that we need to be able to process it. The information contained in the sound signal in DRM is organized in socalled frames with a duration of 400 ms. We need at least this amount of recorded sound to be able to decode some data from it. If the recording is not synchronized with the frame we need even more.

In **drmtst** this problem is tackled by providing an extra layer of buffering between ALSA and the processing of the recorded sound. This bufffering should take place at the moment that ALSA's internal buffer has not yet overflowed and during the time that the processing of the last completely received frame can still be in full swing. The ALSA sound system provides a way to **signal** that its internal buffer needs attention. Via the operating system Linux this signal can be used to trigger an **asynchronous** handler routine. In **drmtst** this asynchronous signal handler fills a (larger) buffer in **drmtst** itself by interrupting the work it was doing. As soon as the transfer of sound samples from ALSA's internal buffer to **drmtst** is completed, **drmtst** resumes the work it was doing.

Synchronization between ALSA and **drmtst** now is simple: **drmtst** only has to check whether there is enough recorded sound in its own buffer. If not, it should wait until there is. On the other hand, if **drmtst**'s own buffer overflows before it is emptied, the processing of the former data has taken too long. If the latter situation arises, stopping other tasks running on the computer may help; if not, the only thing to do is to look for a faster CPU.

### 2.1.2 generation of waterfall picture

The incoming audio is sampled at 48000 Hz. To construct a waterfall display of its frequency content, fast fourier transform (fft) is used after lowpass filtering and decimation the signal by a factor 4. Using a 2048 point fft then gives a frequency resolution of 11.7 Hz. With a window of 256 pixels a waterfall comprising 0 to about 3000 Hz of the signal spectrum can be displayed using the first 256 points of the fft.

The audio data is acquired in bursts of 400 ms. The waterfall display is refreshed in this rythm, so it will appear somewhat "jerky". This is inherent to this simple setup.

### 2.1.3 demodulation

For the demodulation the DIORAMA polyphase filters are used. As these operate on a 12000 Hz IF signal, the audio acquired in step 1 must be converted to a USB signal at an IF of 12000 Hz. The ARRL Handbook [7] shows how this can be done by using a Hilbert transformer in a half complex mixer.

The FIR-filters for the Hilbert transformer in **drmtst** were designed with the use of MATLAB. The design was started with a 83 tap low pass filter with a passband frequency of 3000 Hz and a stopband frequency of 3300 Hz. This lowpass filter was shifted to a bandpass filter with a passband from 200 to 3200 Hz by multiplying with a complex exponent at 1700 Hz. The resulting in-phase and quadrature filtercoefficients are incorporated in the program in the routine **monorec.c** in the variables $B\_Inphase$[83] and $B\_Quad$[83] respectively.

The remaining steps in **drmtst**, i.e. channeldecoding and sourcedecoding are a straightforward translation of the DIORAMA code. For their description see the DIORAMA papers.

## 2.2 The Graphical User Interface

The GUI is implemented in Tk/Tcl in the script **rxamadrm.tcl** and is shown in figure 1. The setup of this script is fairly straightforward. It starts by loading a number of required packages, i.e. Tk, Expect and Img. Then follows the definition of a number of subroutines:

- set_snd - sets the ALSA number of the sound device to be used to acquire the input signal
- file_admin - sets a number of variables connected with the name of the file to be stored with the (picture) data
- rsdecode - spawns the correct routine for reed/solomon decoding of the file that was received
- dispingui - sets the variables that govern which picture file will be displayed in the GUI
- cleanup_incomplete - removes files with invalid picture data
- dispfile - reads the extension of the received file and takes the correct measures to display it if possible

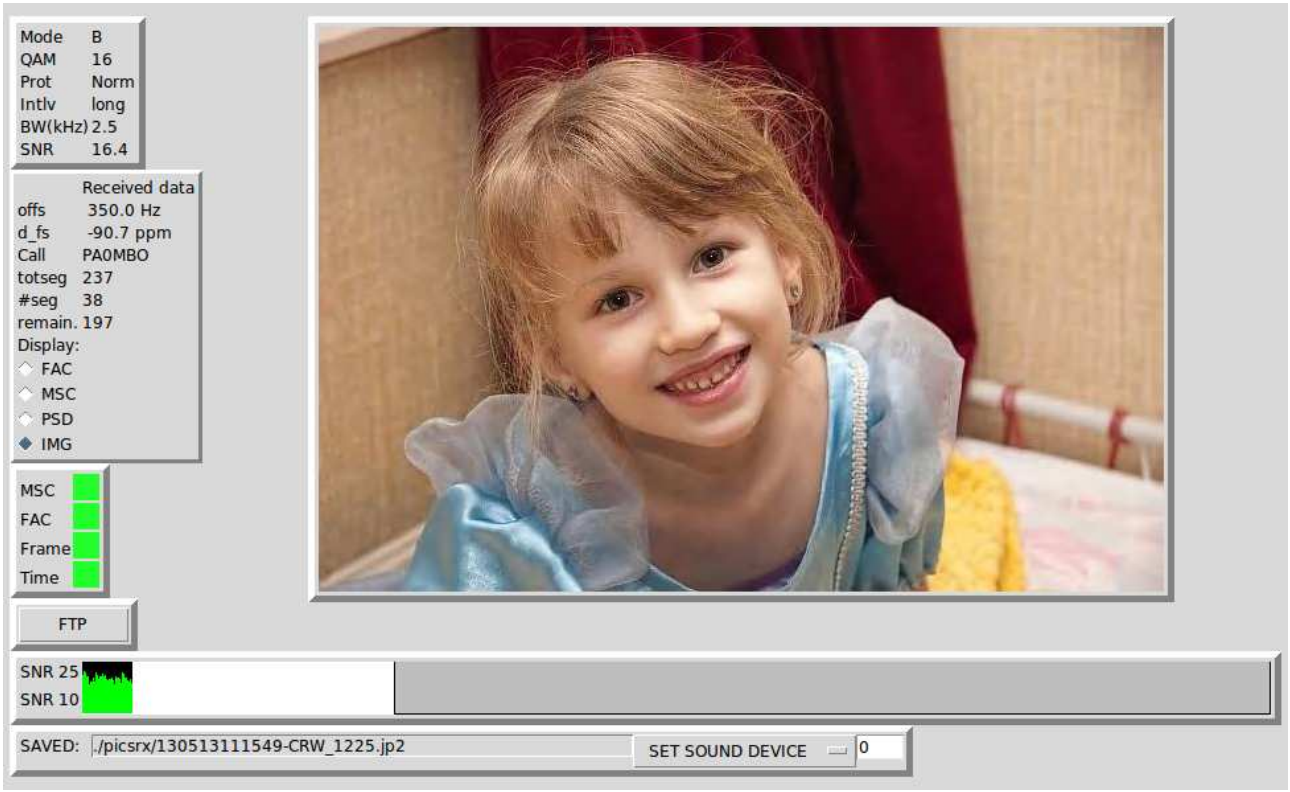


Figure 1: GUI in picture mode

Next the layout of the GUI is defined using frames and various Tk widgets. After spawning the **drmtst** program an endless loop is entered in which **expect** interprets the data sent to stdout by this program and changes the contents of the various widgets accordingly. When files with the reed/solomon extensions rs1, rs2, etc. are received, the correct decoding program is spawned before trying to display the data as a picture.

## 2.3 Reed/Solom decoding programs

The Reed/Solomon decoding programs **rs1decoderas, rs2decoderas, ...** use routines developed by Phil Karn (KA9Q) [8]. These programs use information about missing segments in the received file which is saved by **drmtst** in a diskfile called **erasures.dat** from which the positions of the erasures in the data blocks are calculated. The use of this information doubles the number of errors that can be corrected.

Table 1 shows the parameters for the four Reed/Solomon decoders. For the offline decoding of Reed/Solomon coded files the programs

Table 1: Reed/Solomon parameters

| type | block length | data length |
|---|---|---|
| rs1 | 255 | 224 |
| rs2 | 255 | 192 |
| rs3 | 255 | 160 |
| rs4 | 255 | 128 |

**rs1decode, rs2decode**, etc. can be used which do not need the missing segment information. However these programs can correct only half as much errors.

## 2.4 Reception of EasyPal Hybrid mode pictures

Since August 2013 the Windows program EasyPal offers the use of a socalled hybrid mode in which only a the name of a picture file is sent by radio. To display this file it must be downloaded from a central fileserver. **RXAMADRM** automates this process using the Tk/Tcl-script **hybridget.tcl**.

# 3 INSTALLATION

**RXAMADRM** is now part of the archive **trxamadrmv3_5.tgz**. For installation instructions see the document **trxamadrm.pdf**.

# 4 USING RXAMADRM

Connect the audio output of your transceiver to the mike input channel of your soundcard, start your receiver and tune it to a SSTV frequency using the correct sideband. Assuming the software has been started as described in the document **trxamadrm.pdf** with the command ./startdrmtrx.sh you will see the GUI as shown in figure 1 together with a window that shows a waterfall picture of the audio passband of your transceiver.

To start receiving pictures make sure you set the right sound device with the button at the bottom right of the GUI. This setting will be retained in a file called **rxamadrm.ini** which is placed in your $HOME directory. Now left click your mouse on the radiobutton labeled "PSD". A graph called "power spectral density" will appear (see figure 2). Adjust the audio out of your transceiver to give a top somewhat above the horizontal crossbar. Wait for a DRM SSTV signal to come in and fine tune it by aligning the three bright frequency pilots in the waterfall with the three black vertical markers at the top of the waterfall window (see figure 3 ). Depending on the quality of the signal you will now see the synchronization bars Time, Frame, FAC and MSC turn green. As soon as FAC reaches a steady green the data displayed in the upper left block and the top three items of the middle left block will get sensible values.

The top block shows the characteristics of the received signal:

- Mode - the mode of the drm signal: A, B or E
- QAM - the number of modulation states: 64, 16 or 4
- Prot - the protection level: normal or high
- Intlv - the interleaving distance: long, short
- BW(kHz) - band with of the received signal
- SNR - the Signal to noise ratio in dB

The middle block shows the offset of the DC-carrier of the signal (should be around 350 Hz), the relative error of the sampling frequency of the ADC in the soundcard in ppm and the callsign of the sending station.

When the MSC item turns to a steady green, segments are being decoded successfully. When a complete header has been found the

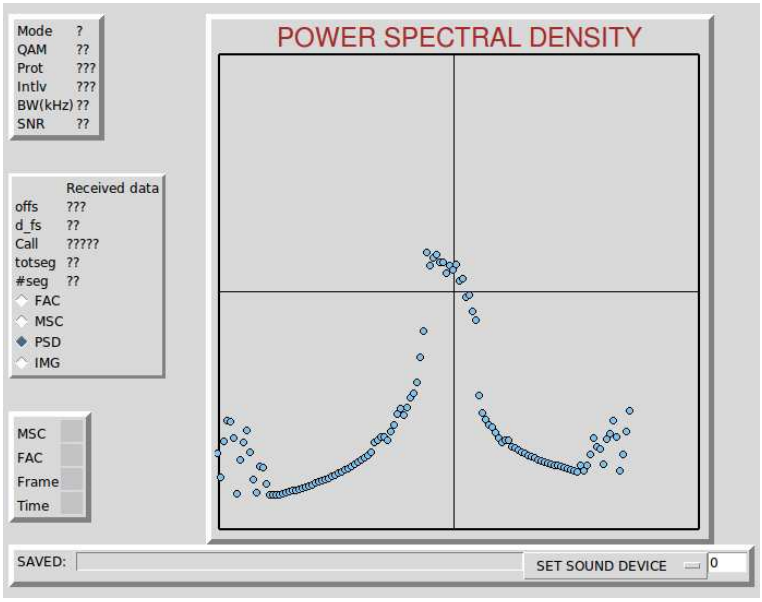


Figure 2: GUI showing PSD with correct audio level

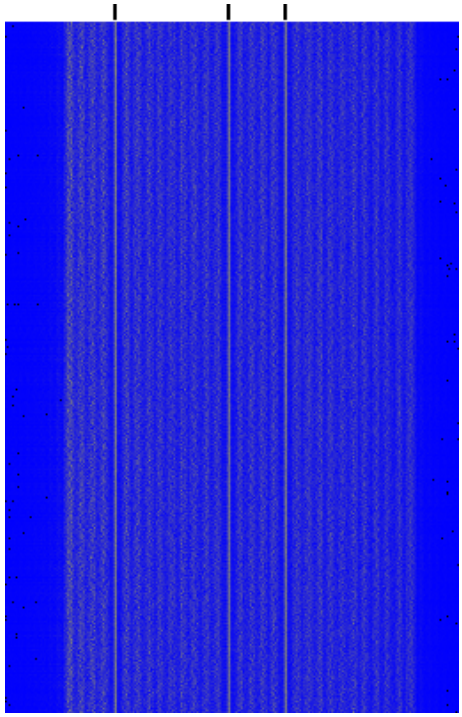

Figure 3: Waterfall window with DRM signal

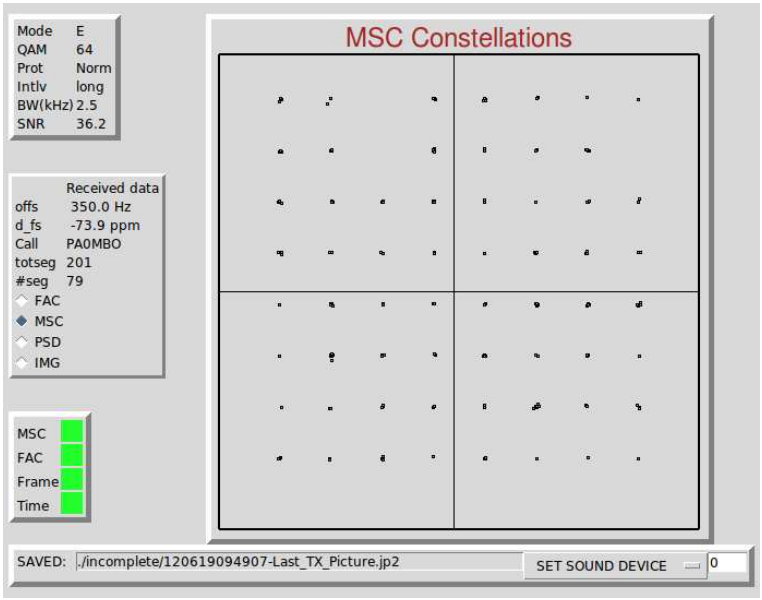


Figure 4: GUI showing MSC-constellation

item **totseg** shows the total number of segments of the file that is being transmitted. The item **#seg** shows the number of the last segment that was received correctly. When things go smoothly you will see this item increase steadily.

The radiobuttons at the lower half of the middle block at the left side of the GUI allow you to choose what will be displayed in the right handside window of the GUI

- FAC - the constellation of the Fast Access Channel Data
- MSC - the constellation of the Main Service Channel Data
- PSD - the power spectral density of the received signal
- IMG - the last image that was received correctly

FAC and MSC will only be active when their corresponding sync sign is green. The MSC displays is given in figure 4. When you see such a regular dot pattern the quality of the signal is perfect, but you will be surprised which irregularities in the MSC constellation patterns still give good copy. At the bottom left of the GUI there is a box showing the name of the last file that was received, either completely correct or else incompletely. In the first case the file will be in the **pics** subdirectory, in the second case it will be in the **incomplete** subdirectory.

The dropdown button labeled "SET SOUND DEVICE" at the lower right of the GUI allows to set the ALSA number of the soundcard that is to be used for the input of the transceiver audio.

## 4.1 FTP

The GUI of **RXAMADRM** has a button labeled "FTP". When pressed it opens an entry form with a number of buttons. Use the entry boxes to give the details for ftp-ing your provider, i.e. your username and password, plus the name of the server that you want to address. Depending on the organization of your website you will also have to provide the name of the directory of the page where you want to upload the images. The button "xfer method" opens a pull down menu with choices between normal ftp, secure ftp and secure copy. Choose the one that is supported by your provider. If you want something more sophisticated it is not too difficult to change the script **ftpput.tcl** in the directory **trxamadrmv3_5/linux**. It shows lines with the prompt to be expected from your providers server followed by the commands to be given in sequence.

After pressing the buttons "SET FTP ON" and "CLOSE" the FTP window disappears, but when a picture has been received successfully it will be uploaded to your website automatically. The information will be stored in a file called **ftp.ini** and is placed in your $HOME directory. Be careful in standalone use of this feature, as you never know what kind of pictures will be uploaded. Manual upload can be accomplished via the button "UPLOAD LAST" in the FTP window.

## 4.2 Progress status bar

As of version **trxamadrmv3_2** the GUI has a bar showing the progress of the reception of a picture. Initially this bar extends over the full width of the rxamadrm windown. As soon as the total number of segments for the file under reception has been established the bar is delimited at the right handside to the correct size. The signal to noise ratio for a segment that was received successfully is show by the height of a vertical green line ranging from 5 dB at the bottom of the bar to 25 dB at its top. Missing segments show up in this bar as white gaps. The complete bar is whitened out when the start of a new file is detected. For fixes gaps are filled in and parts of the graph are overwritten.

## 4.3 Log files

When started as described above **rxamadrm** will produce two separate logfiles, i.e. **dbgrxama** and **rxlog.txt**. The first contains all info that was produced by **drmtst** as well as all output produced by the GUI and can be used to trace all events. For normal use **rxlog.txt** is

far more useful and shows successful names of received and decoded files and SNR data both with date/time stamps.

## 4.4 Hybrid mode

To be able to receive pictures sent by EasyPal in the socalled hybrid mode you have to switch on this mode by pressing the button with the label "HYBRID" and fill out the form that appears and press its "SET HYBRID ON" button. For interacting with stations using EasyPal you have to enter the access data for the general ftp-server of VK4AES or agree upon a particular server to use with the other station(s). The data are also used in sending "hybrid" pictures. The text of this buttons changes to "SET HYBRID OFF" after which you can close this form by pressing its button "CLOSE". It you want to switch off this mode later press the button "HYBRID" in the main menu again and press the button "SET HYBRID OFF" and close the window again.

## 4.5 Computer to computer testing

During the development of RXAMADRM the audio signal generated by DIGTRX running on a second computer was used. The audio line output of the soundcard in the computer running DIGTRX was connected to the microphone input of the Linux system running RXAMADRM. All DRM-modes A,B and E were checked with both settings of interleave (long or short), with both settings of protection (Normal or Low) and all possible QAM-modes. SNR was over 38 dB in this set up. Transmitted files were jpg's. Under all circumstances the reception was without missing segments. A screenshot of the reception of a 50 kB file in mode A with QAM-16 is given in figure 1.

## 4.6 Testing on the 80 meter band

Figure 5 is the first picture I received using RXAMADRM on the band at 10 dB SNR. It was a Reed Solomon coded file 090301202608-Clip.rs2 of 14025 bytes from the station G8ITH.

Figures 6 and 7 show some more received pictures. I have no data on SNR because they were received in the standalone mode of **drmtst**. I just found them in the **./pics** directory after I left the program on for some time.


Figure 5: Picture sent by G8ITH


Figure 6: Picture sent by DF2L on 80 m on March 4th 2009


Figure 7: Picture sent by DF2ML on 80 m on March 5th 2009

# 5 DISCUSSION

The main reason to develop RXAMADRM was to provide the Linux community of hams with a good starting point to develop their own drm-based digital SSTV software. Linux offers a large variety of GUI-builders of which Tk/Tcl is just one. With toolkits like Gtk+ it is possible to build very sophisticated screens with attractive readouts, display of images and controls. Especially when these toolkits are combined with scripting languages they make it easy to automate many tasks like logging, web publishing, generating e-QSL's etc. RXAMADRM offers all required real-time data-acquisition and processing to convert the lf audio output of a receiver tuned to a ham DRM SSTV station to a diskfile of the transmitted image that is an exact copy of the one the sending station used.

The **rxamadrm.tcl** script shows an example of how this functionality can be "dressed" with a graphical user interface that can be customized to one's personal liking. It also allows to supplement the system with convenience functions like logging, archiving of pictures, etc.

# 6 ACKNOWLEDGEMENTS

The author is very much indebted to David Ranch (KI6ZHD). Without Davids help **rxamadrm** would still be a barebones DSP program

for decoding DRM SSTV. Its current features, i.e. Reed/Solomon decoding, display of .txt-files, the waterfall, the possibility to set the sound input device from the GUI, etc. are not only all his ideas, he also tested my attempts to implement these features until they worked. (Remaining bugs are of course my responsibility!). I also want to thank ON4QZ, who has used part of **rxamadrm**-code into his fine program QSSTV and tested **rxamadrm** very thoroughly. I hopefully corrected the errors he found without introducing new ones. Dr OM Johan many thanks.

# References

[1] Cesco HB9TLK, http://www.qslnet.de/member/hb9tlk/.

[2] http://n1su.com/windrm.

[3] VK2CZU, http://www.users.on.net/ trevorb/.

[4] http://www.tima.com/ djones/hamdrm.htm.

[5] T. Schorr A.Dittrich, http://nt.eit.uni-kl.de/forschung/diorama.

[6] T.Schorr A.Dittrich W. Sauer-Greff R.Urbansky, http://www.fh-kl.de/ drm/dokumente/sonstige/ieee_sp2005_diorama.pdf.

[7] R. Dean Straw, editor. *The ARRL Handbook For Radio Communications*. ARRL, 2006.

[8] Phil Karn, http://www.ka9q.net.

# APPENDIX

## LIMA-SDR

For use with the LIMA-SDR a special version of **rxamadrm** has been developed which is called **rxlimadrm**. This version samples the I/Q - signals generated by the Lima-receiver instead of the normal audio signal from a non-SDR receiver. For this a stereo-input of the soundcard is needed, i.e. the line-input. Moreover the software now also has to tune the receiver using the correct sideband. The LIMA-SDR is controlled by an AVR/USB-processor that connects to the USB-bus of the computer. A separate program **usbio2** has been developed for the task to address the AVR/USB-processor and the GUI has been extended with an entry box for setting the rx-frequency and two radio buttons for the sideband choice.

To compile **rxlimadrm** your linux system now needs the **libusb-dev** package installed. When using **rxlimadrm** not being root, some special measures are necessary to allow access of the USB-device of the LIMA-SDR. The following steps describe how to obtain this permission:

```
cd  /rxlimadrm
su
give your root password
chown root usbio2
chmod u+s usbio2
```

## Using rxlimadrm

Change directory to the subdirectory **rxlimadrm** and give the command:

```
./startlima
```

Set the frequency of the DRM-SSTV channel you want to receive in the entry box at the right hand side of the button labeled "SET QRG" and make your choice between lsb and usb via the radio boxes labeled as such. Then press the button "SET QRG" and the reception will start at the chosen frequency.

While running **rxlimadrm** qsy can be established by changing the frequency in its entrybox and pressing the button "SET QRG", just changing the numbers will **not** send the changes to the LIMA. Changing the chosen sideband will automatically invoke a change of the settings of the LIMA-SDR (and a change of the DSP-filters).