INTERNAL REPORT

AN RFI MONITORING SYSTEM BASED ON A WIDE-BAND DIGITAL BACK-END FOR THE SARDINIA RADIO TELESCOPE

A. Melis*, R. Concu*, A. Trois*, C. Migoni*, R. Ricci, M. Bartolini** & SRT Astrophysical Validation Team(#)**

Report N. 36, released: 21/07/2014

Reviewer: G.P. Vargiu

OAC Osservatorio Astronomico di Cagliari

Abstract

One of the major issues in radioastronomy beyond a shadow of a doubt are the RFI. Several solutions can be adopted to reduce their impact, among them digital back-ends operating in piggy-back mode on the signal coming from the telescope.
In this report we will describe an infrastructure hardware/software developed for the Sardinia Radio Telescope. The system act as a real-time monitoring exploitable by astronomer in site during an observation but it can be very helpful for a dynamic scheduling as well . Finally, we will show how the RFI detection algorithm employed at the Arecibo radio telescope can be applied and exploited for the SRT.

(#) continued from cover page:

R. Ambrosini**, P. Bolli***, M. Burgay*, M. Buttu*, P. Castangia*, S. Casu*, G. Comoretto***, A. Corongiu*, N. D'Amico*, E. Egron*, A. Fara*, F. Gaudiomonte*, F. Govoni*, D. Guidetti**, N. Iacolina*, F. Massi***, M. Murgia*, F. Nasir*, A. Orfei**, A. Orlati**, A. Pellizzoni*, D. Perrodin*, T. Pisanu*, S. Poppi*, I. Porceddu*, I. Prandoni**, A. Ridolfi*****, S. Righini**, C. Stanghellini**, A. Tarchi*, C. Tiburzi*, V. Vacca****, G. Valente*, and A. Zanichelli**.

** INAF - Osservatorio Astronomico di Cagliari*
*** INAF - Istituto di RadioAstronomia di Bologna*
**** INAF – Osservatorio Astrofisico di Arcetri*
***** Max Planck Institut fur Astrophysik, Garching, Germany*
****** Max Planck Institute fur Radioastronomie, Bonn, Germany*

1 Introduction

Radioastronomical observations have to take into account different variables. Usually, in order to ascertain that each component is working as it should, one or more Tsys are calculated.

Nevertheless, a real-time spectral information of the observed bandwidth represents the best way to be sure that the entire receiving chain is working properly. The idea to implement an infrastructure controlling the signal bandwidth being observed especially for RFI monitoring purposes has motivated this work.

The infrastructure essentially consists of a wide-band FFT spectrometer operating in piggy-back mode on a copy of the radioastronomical signal, and a Linux-based PC containing the software used for this application.

A QT-based program (described in chapter 2) continuously communicate with the spectrometer and with the control software of the antenna, getting both the spectra and information concerning the telescope status (azimuth, elevation, local oscillator etc). Data are then saved and stored in FITS format (see chapter 2) and after that an RFI detection algorithm developed for the Arecibo observatory is applied, as described in chapter 3.

2 Hardware and software description

In this chapter we will explain how the infrastructure works step by step. After a brief description of the digital spectrometer, the infrastructure including GUI and FITS data converter will be shown.

2.1 DBBC scansion spectrometer

The spectrometer is implemented on the Digital Base Band Converter [1], i. e. the new digital VLBI terminal SRT is equipped with. Usually, in radioastronomy a spectrometer has a conventional configuration, with a single stage polyphase filter bank used to split up the entire input bandwidth in smaller pieces and whereupon each channel is squared and summed up for an integer number of FFT cycles. Due to the possible

strong emission like radar or other kinds of radio interference, the FFT engine could saturate and the entire spectrum irretrievably goes bad. The solution adopted in this case was to make a two-stage polyphase filter bank [2], like shown in figure 1. The first [3] polyphase filter bank divides the digital input with a 50% overlapping (see fig. 2 & 3) in order to avoid spectral holes into the input signal, while the second ones process couple of signal from the previous stage.

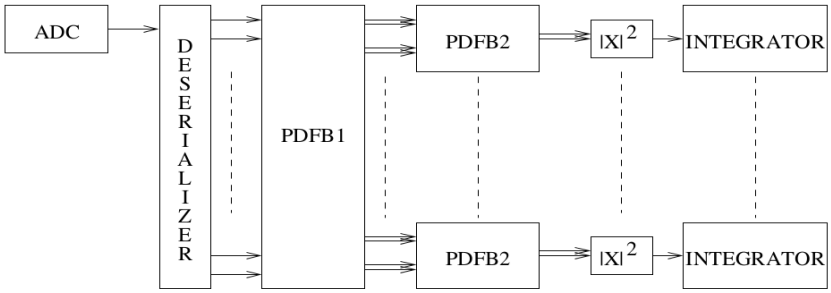

Figure 1: Two stage polyphase filter bank spectrometer

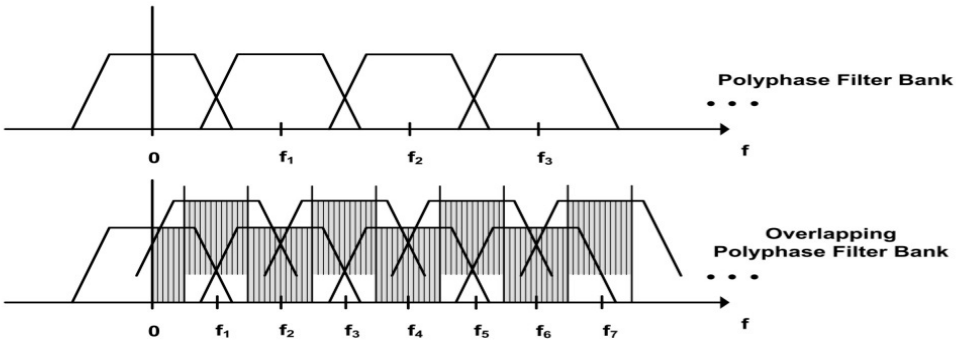

Figure 2: First stage polyphase filter bank

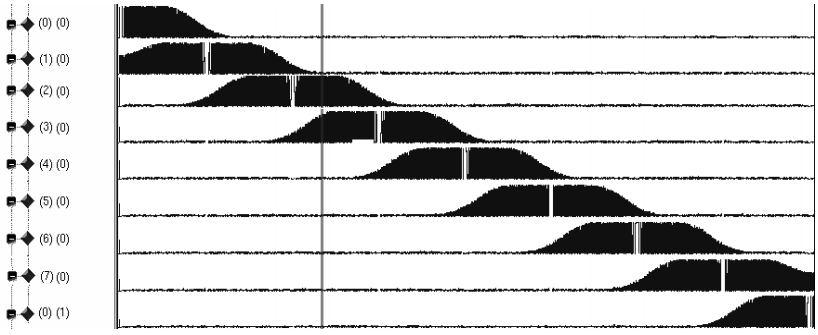

Figure 3: Simulation of the first polyphase filter bank with a swept sinusoidal signal

Such a particular configuration makes the spectrometer stronger from an RFI point of view. The specificity of the design also permits saving FPGAs resources more than a traditional overlapping case, especially BRAM. Figure 4 shows an example with an 8-point FFT engine in which it can be seen that just half memory is necessary to store the spectra because the flat part of each stream is automatically got, throwing away the “bad” zones.

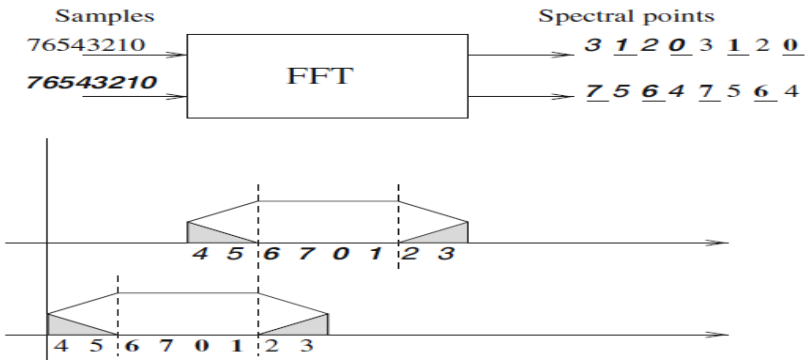

Figure 4: Automatic selection of the flat band (8-point FFT example)

2.2 QT-based software infrastructure

The control software of the infrastructure has been written in C++ language, it uses a QT 4.8 based framework. Figure 5 shows the application GUI. Essentially it interrogates the antenna's control software [4], called NURAGHE, via a dedicated server named “*external client*” that merely makes possible retrieving informations like azimuth, elevation, local oscillator, etc; most relevant of them also be plotted in the graphical interface.

An example of what NURAGHE provides when an external client request is done is the following:

antennaParameter/2014_190_13:44:26.456,OK,Tsys,162.5989,082.1303,+09:41:19.892,+32:00:56.056,+194:18:31.068,+48:42:10.669,0.000037,0.000039,0.041159,0.001932,0.000040,0.000000,0.000000,0.008639,0.013352,0.000000,0.000000,CCB,6900.000000,TRACKING

A QtcpSocket class is used to establish connection with the NURAGHE server, then the string *“antennaParameter”* is sent to the external client and using the callback

function *void MainWindow::readData()* the received data are parsed and then the GUI is updated.

According with the receiver selected, corresponding button turns on green (L-band in case of fig. 5) and left or right polarization can be chosen. Pushing the *"START MONITORING"* button, apart from the fact that it turns into *"STOP MONITORING"*, a *gnuplot* script keeping account of the previous selection is dynamically created and started up via a process using a Qprocess class. In order to kill the process, the button must be pushed again.

The *"Save Data"* option must be activated to store the data; in the next paragraph every detail will be outlined.

In addition, the *"Send Data to"* option allow sending spectra and antenna parameters informations to a client specified with the corresponding Internet Protocol address.

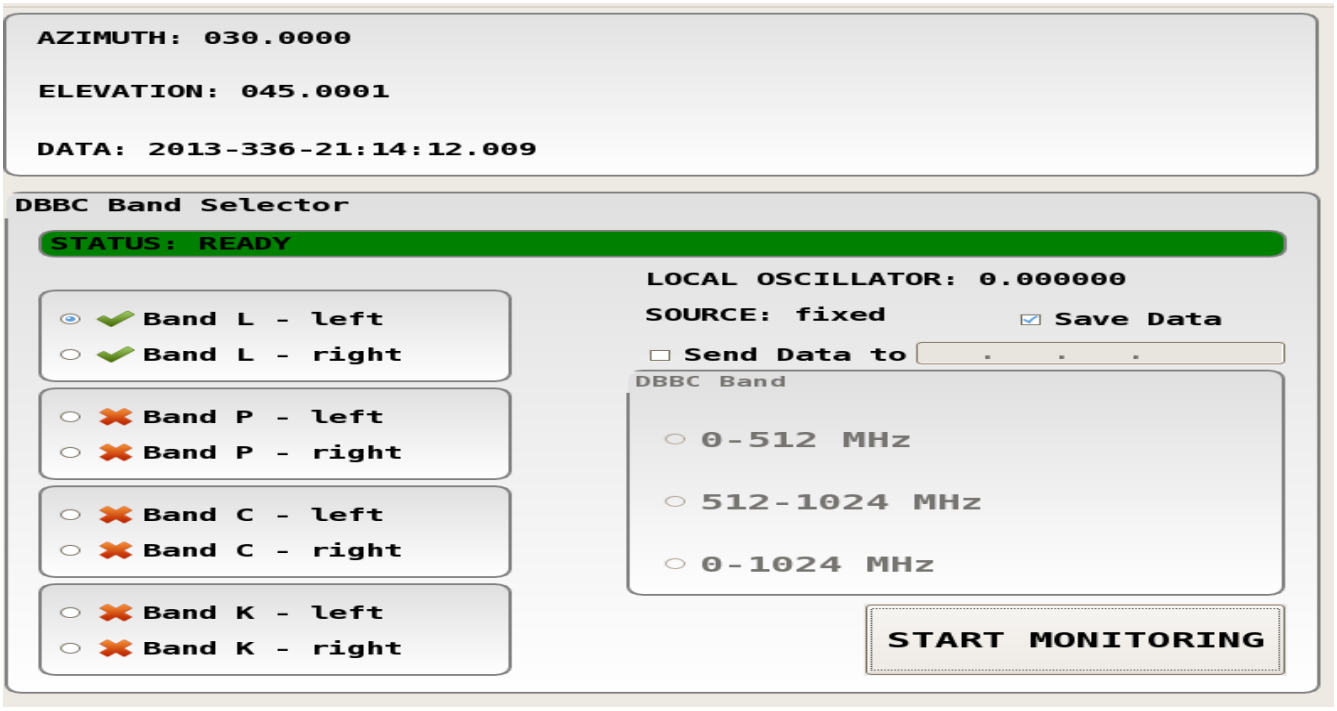

Figure 5: Application's GUI

2.3 FITS data converter

The FITS data converter rewrites the data acquired by the control software in more suitable format for astrophysics purposes.

The input file is in a TXT format in which the spectrum from the DBBC follows the string showed in the previous paragraph.

No information is added, just a translation/reorganization of the available data is done; the output files is the well known FITS standard format.

The task can be run and checked by means of the following command (see fig. 6):

~/bin/dbbc_fits_writer par1 par2 par3

where:

par1 is the complete path and file name of the txt input

par2 is the complete path and file name of the fits output

par3 is the the override parameter (1 means new file)

```
alessio@arc3:~/DBBC/DBBC_Monitor_RFI> ~/bin/dbbc_fits_writer $PWD/DATA/2014189/TXT/dbbc_20141891155554.418_0.000024_
180_044_fixed.txt $PWD/DATA/2014189/dbbc_201418914.fits 0


START EXECUTION:  Mon Jul 14 12:41:16 2014
#####################################################
#########  Task dbbcfitsConverter......... starting  #########
#########         version 1.1 06/06/2014         #########
#####################################################


dbbcfitsConverter..................... starting OpendbbcfitsConverter
dbbcfitsConverter..................... exiting OpendbbcfitsConverter STATUS = 0

dbbcfitsConverter..................... starting ClosedbbcfitsConverter
dbbcfitsConverter..................... exiting ClosedbbcfitsConverter STATUS = 0



#####################################################
#########  Task dbbcfitsConverter........... exiting ##########
#####################################################


STATUS 0
```

Figure 6: Start and execution of the FITS data converter

The DBBC FITS file has the structure shown in figure 7. The FITS data converter's task with which data are written into the archive is executed through a pipeline activated by the control software.

The data archive structure is based on high-level directories corresponding to the year and to the day following this name convention:

.../DATA/2014189/FITS
.../DATA/2014189/TXT
.../DATA/2014190/FITS
.../DATA/2014190/TXT
.../DATA/2014191/FITS
.../DATA/2014191/TXT
.....
The DBBC FITS file names has the following structure:

dbbc_YYYYDDDHH.fits
where:
YYYY = year
DDD = day of the year
HH = hour of the day

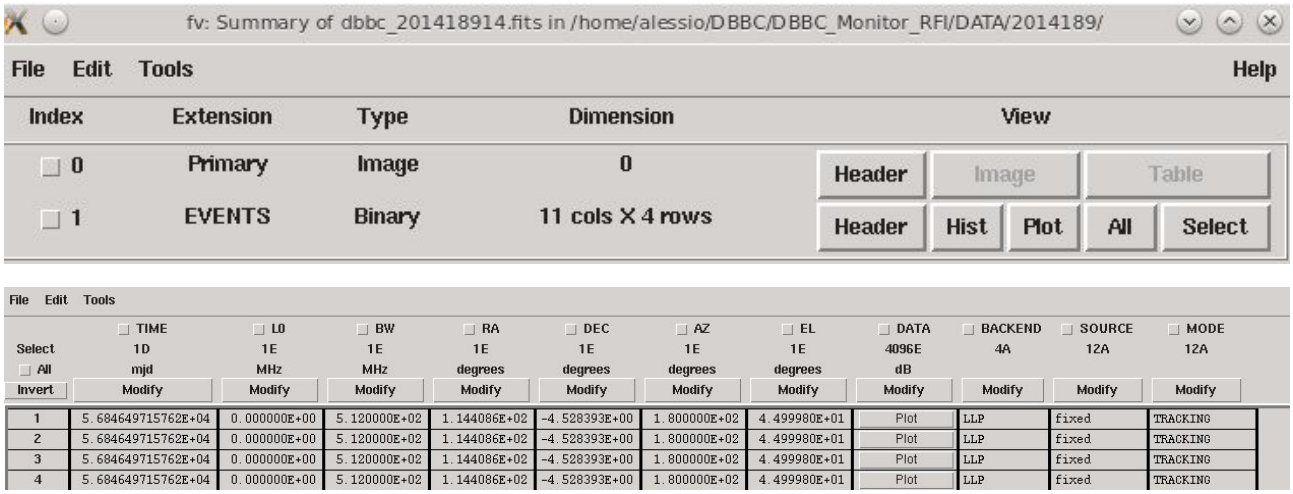

Figure 7: DBBC FITS file format

3 RFI detection algorithm

Once the data are retrieved, an RFI detection algorithm can be applied. Firstly we will describe the necessary data format conversion applied on the data, then we will illustrate how the algorithm acts to identify whether we are dealing with interferences.

3.1 Conversion to GALFACTS formats

The GALFACTS [5] RFI detection pipeline works in combination with its own data format as input. Data acquired from DBBC and afterwards stored in *.fits* files must thus be converted into the proper format, i. e. a binary file in *.spec* format containing actual data and some ancillary information stored in a text *.spec_cfg* file associated with it.

3.1.1 SPEC and CFG files

We have analyzed some files acquired at Arecibo along with a look at some of the sources of the pipeline; some preliminary considerations about this can be viewed in the form of an ipython notebook [6].

The **spec** file consists of a header and a payload. The header contains pointing informations of the central beam and six outer beams, while payload contains X and Y polarizations in all their possible cross-combinations with calibration mark on and off respectively; this makes eight combinations possible: xx-on, xx-off, yy-on, yy-off, xy-on, xy-off, yx-on, and yx-off. Note that *spec* files is made up for acquisitions of 4096 frequency channels per polarization and this seems to be hard-coded in the pipeline; eventual changes in the number of bins should be tested on the whole pipeline and may not be possible without a greater effort.

We have defined a mapping between the central beam header of the *spec* file and the informations saved in the *dbbc* file as follows; each row of the *fits* file will have:

Spec header	**DBBC data**
raj_true_in_hours	data_row[“RA”] / 15.0
decj_true_in_degrees	data_row[“DEC”]
epoch	2000.0 (fixed value)
atlantic_solar_time_now_in_seconds	mjd2ast(data_row[“TIME”])
az_cur_in_degrees	data_row[“AZ”]
za_cur_in_degrees	90.0 - data_row[“EL”]

Outer beams headers are empty because not necessary, while actual data are copied equals in each polarization's combination in order to preserve binary compatibility with the original format. Note that this will make the *spec* file ~8 times larger (in bytes size) than the original *fits* file.

Successive rows from the *dbbc* fits file are just appended with their relative header and data sections in the *spec* file, following the same convention.

The **cfg** file is composed by information taken from the first row of the data, together with some hard-coded values as follows:

CFG values	**DBBC data**
Integration (ms)	3000 (fixed value)
MJD	data[“TIME”]
Center Freq (MHz)	data[“LO”] + data[“BW”] / 2.0
Channel band (kHz)	data[“BW”] / channels
Channels	len(data[“DATA”])
Ra at start	data[“RA”]
DEC at start	data[“DEC”]
UTC at start	mjd2ast(data[“TIME”])
ALFA at start	0.0 (fixed value)
Project ID	“DBBC” (fixed value)

3.1.2 Dbbc2spec software tool

Once we obtained the necessary information, we developed a tool that, automatically, converts a *.fits* file acquired using the DBBC spectrometer into a couple of *.spec* and *.spec_cfg* files; this is preparatory for the GALFACTS elaboration pipeline.

The package provides a shell command **dbbc2spec** that takes, as input, a *.fits* file and then generates the relative *galfacts* files. The software is developed using the python [7] programming language, and depends on the package **astropy** [8] that can be gotten at their homesite or installed via pip. At the moment of this writing, *astropy* version is 0.4 .

The software can be downloaded from [9], and can be easily installed with standard methods.

3.1.3 Installation and usage

The following steps document how to install and use this simple tool:

```
($ pip install astropy )

$ wget http://www.ira.inaf.it/~bartolini/rfi/dbbc2spec/dbbc2spec-1.0.tar.gz

$ tar xzvf dbbc2spec-1.0.tar.gz
```

```
$ cd dbbc2spec-1.0/

$ ls

dbbc2spec.py  LICENSE  PKG-INFO  scripts  setup.py  tests

 $ nosetests -v

test_get_params (test_dbbc2spec.TestFileConversion) ... ok

test_make_cfg (test_dbbc2spec.TestFileConversion) ... ok

test_make_spec (test_dbbc2spec.TestFileConversion) ... ok

test_mjd2ast (test_dbbc2spec.TestFileConversion) ... ok

test_sepc_pointing_size (test_dbbc2spec.TestFileConversion) ... ok

test_spec_data_size (test_dbbc2spec.TestFileConversion) ... ok


----------------------------------------------------------------------

Ran 6 tests in 1.140s


OK

$ python setup.py install #you may need to be root

running install

running build

running build_py

creating build

creating build/lib.linux-x86_64-2.7

copying dbbc2spec.py -> build/lib.linux-x86_64-2.7

running build_scripts

creating build/scripts-2.7

copying and adjusting scripts/dbbc2spec -> build/scripts-2.7

changing mode of build/scripts-2.7/dbbc2spec from 644 to 755

running install_lib

...
```

```
running install_scripts
...
running install_egg_info
...
$ dbbc2spec --help
usage: dbbc2spec [-h] [-d] dbbc_fits_filename.fits

convert a fits file as obtained by the dbbc spectrometer into a coule .spec
and _cfg files processabel with the GALFACTS pipeline

positional arguments:
  dbbc_fits_filename.fits
                        path to a dbbc .fits file

optional arguments:
  -h, --help            show this help message and exit
  -d                    set output to debug mode
$ cd tests
$ dbbc2spec dbbc.fits
INFO: opening fits file
INFO: getting dbbc parameters
INFO: writing cfg file
INFO: writing spec file
INFO: OK
$ cat dbbc.spec_cfg
3000
4096
256.000000
```

```
512000.000000

1 4096 4 2 0

DBBC

56628.750011

AO

Integration time (ms): 3000.000000

MJD: 56628.750011

Center freq (MHz): 256.000000

Channel band (kHz): 125.000000

Number of channels/record: 4096

RA at start (degrees): 340.457367

DEC at start (degrees): 9.897650

UTC at start (seconds): 64800.911000

ALFA angle at start (degrees): 0.000000

Project ID: DBBC
```

3.2 GALFACTS RFI detection routine

The G-ALFA Continuum Transit Survey (GALFACTS) team has developed a pipeline able to process the huge amount of spectro-polarimetric data (about 10 TB every 6 observing hours) taken with the Arecibo L-band Feed Array and the Mock FFT spectrometers. The first stage of this data processing pipeline performs a quick look at the raw *CIMAFITS* files taken with the Arecibo telescope. A series of diagnostic plots are thus created on a daily basis to inspect:

- the bandpass shape for each of the seven feeds and the two sub-bands
- The band-averaged total power as a function of observing time
- The winking CAL stability as a function of observing time
- The pointing pattern in RA/Dec
- The RFI situation in each of the seven feed and the two sub-bands via time-frequency RFI flag plots.

All of these quick-look plots are made available to the team members for inspection via a dedicated web interface.

A single routine named *spec2fits* makes all these quick-look stage possible. It is written in C language as the rest of the GALFACTS pipeline. It is made available by the GALFACTS team for the SRT RFI monitoring pipeline and it is installed on the *meteoserv* machine at the SRT. The routine reads a binary file *.spec* holding the radio spectra dumped by the DBBC and an ancillary *.spc_cfg* config file containing additional information used by the routine *spec2fits* to process the radio spectra. When *spec2fits* is launched without arguments, the routine returns the following prompt:

> ./spec2fits

Usage: ./spec2fits <specfilename> <smooth> <numsigma> <numsigmathresh> <ignoreA_low> <ignoreA_high>

An example is the following:

./spec2fits A2174.perpuls_08_00_025346+250905.beam0.53989.spec 1 5 90 875 890

Thus, the routine requires this list of input parameters:

<specfilename> .spec file name preceded by the complete path

<smooth> switch to perform Hanning smoothing : 0 = no; 1 = yes

<numsigma> number of sigmas above which the routine performs RFI rejection

<numsigmathresh> threshold on the number of sigmas above which *spec2fits* performs the RFI rejection (the RFI detection algorithm is described in GALFACTS internal Memo17)

<ignoreA_low> lower channel number in the range of channels to be ignored when the routine rejects the RFI-affected channels.

<ignoreA_high> higher channel number in the range of channels to be ignored when the routine rejects the RFI-affected channels. Set them both to zero for no ignore channel range.

The values of <numsigma> and <numsigmathresh> presented in the example are optimized for the Arecibo data, but they appear to work well for DBBC FITS data, too, as it will be shown in the next section.

The *spec2fits* routine outputs a list of text files:

filename.spec_bandavg.dat => band-averaged total power

filename.spec_noise.dat => winking CAL monitoring

filename.spec_pointing.dat => pointing monitoring

filename.spec_rfi.dat => RFI monitoring

filename.spec_timeavg.dat => time-averaged bandpass shape

diff.dat => statistical output

finalmean.dat => statistical output

finalsigma.dat => statistical output

sigmathresh.dat => statistical output

Of all these output files we only use the *filename.spec_rfi.dat* which contains the following columns:

#chan freq RA DEC AST

1 0.125000 113.499817 -4.527522 42736.89

2 0.250000 113.499817 -4.527522 42736.89

49 6.125000 113.499817 -4.527522 42736.89

50 6.250000 113.499817 -4.527522 42736.89

143 17.875000 113.499817 -4.527522 42736.89

...

Col. 1: spectral channel number between 0 and and 4095

Col. 2: frequency of the channel in base band (when no LO is set) or RF (when the LO is set to a certain frequency by the command setLO in NURAGHE)

Col. 3 and 4: RA and Dec of the sample

Col. 5: time stamp measured in seconds starting from 0h UT. Note: this time stamp is mid-time between two consecutive time marks in the FITS file.

The output file described above is then used to create RFI diagnostic plots such as: (i) time vs frequency RFI flag plots and (ii) fractional RFI occupancy plots as function of

channel number or frequency.

4 Results

We took a series of scans on July 8th 2014 with the SRT equipped with the DBBC in order to test the full RFI monitoring pipeline. The FITS files were the following:

Filename	Az	El	Tot int. time	BW	nch	df	LO
	deg	deg	min	MHz		MHz	MHz
dbbc_201418911	180	45	7.7	512	4096	0.125	1292
dbbc_201418912	0-360	45	60	512	4096	0.125	1292
dbbc_201418913	0	45	6	512	4096	0.125	5600

The DBBC/FITS files were transformed into *.spec* files with the *dbbc2spec* converter and process with the routine *spec2fits.* The *rfi.dat* text files were plotted with *gnuplot* in a time vs frequency RFI flag plot (Fig. 8-9) and compared with the time vs frequency waterfall image of the raw FITS file data (Fig. 10-11). These images are created with the custom-made IDL procedure *plotwater.pro.*

Finally we created the RFI occupancy plots as a function of frequency with the custom-made Perl script *band.pl* and IDL procedure *rfiocc.pro* (Fig. 12-13) .

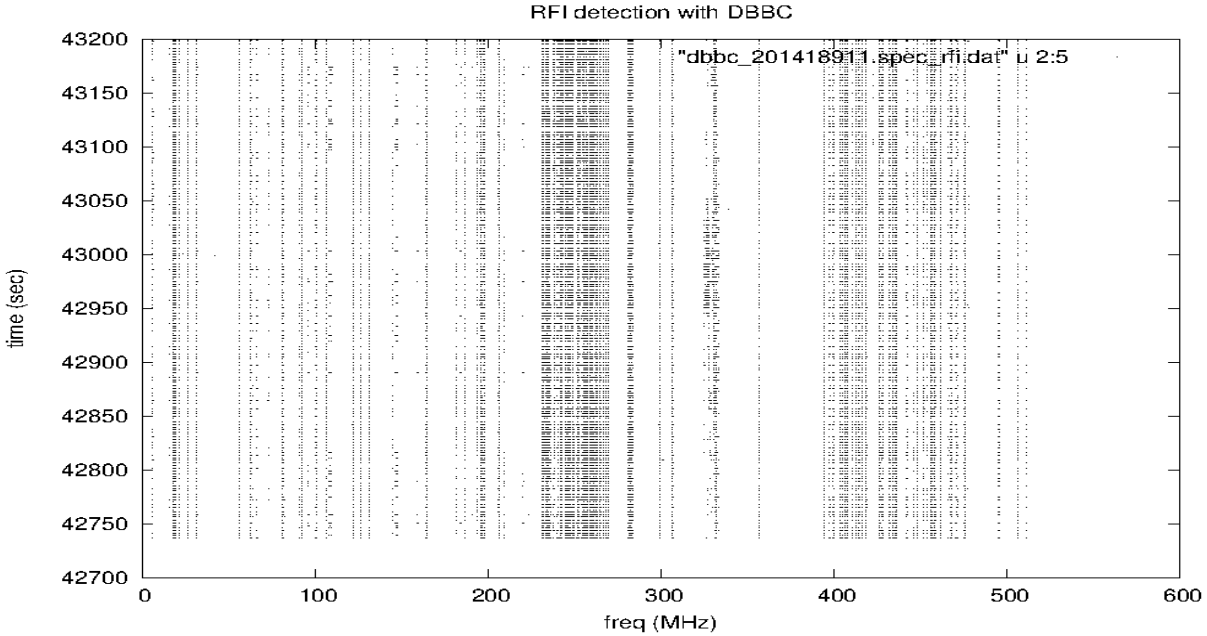

Figure 8: *dbbc_201418911.fits* data (L band) plotted as time vs frequency RFI flag plot. Each dot represents a time-sample/freq-channel flagged as RFI affected by *spec2fits.* The horizontal axis show the base-band frequency.

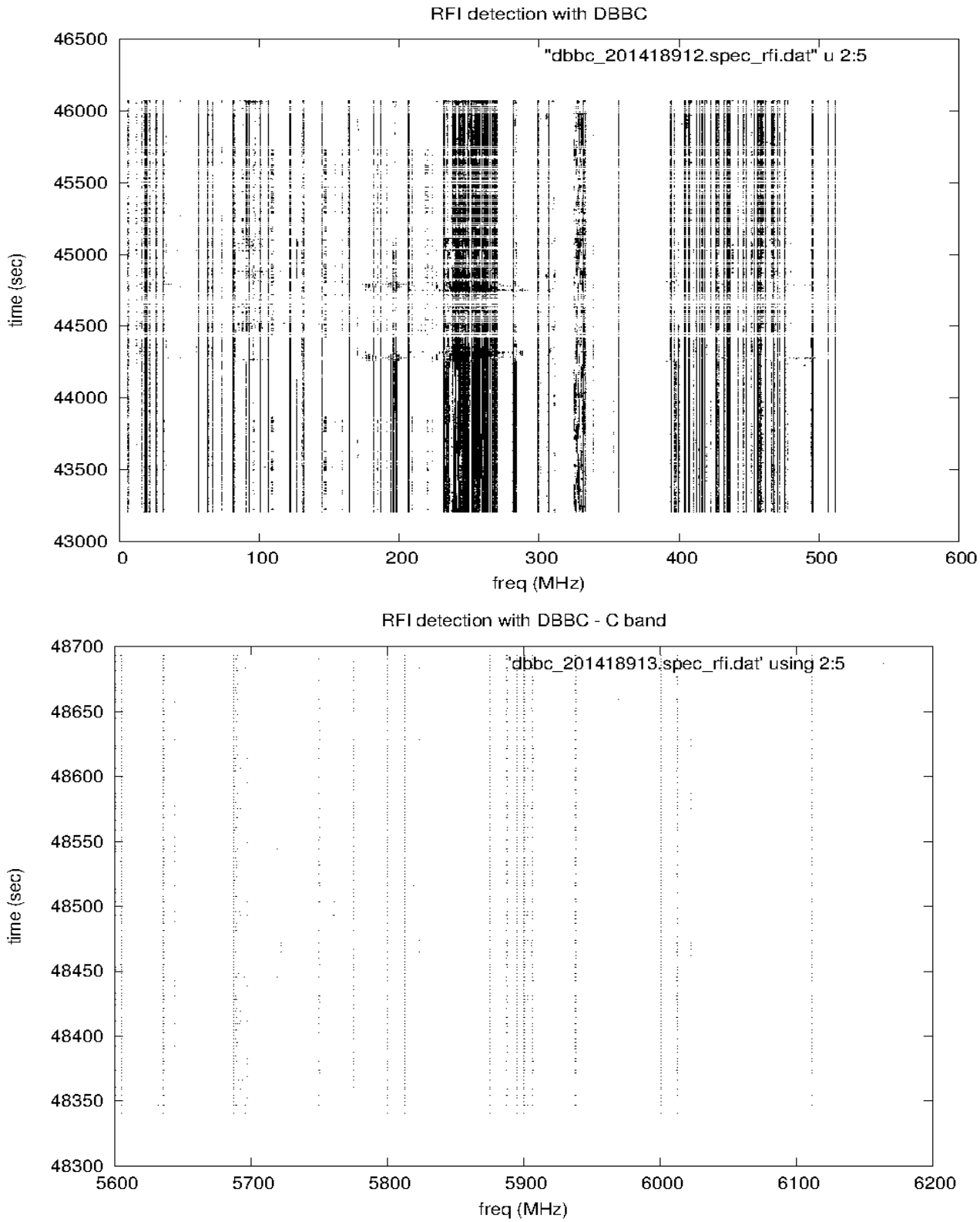

Figure 9: dbbc_201418912.fits (L band, top panel) and dbbc_201418913.fits (C band, bottom panel) data plotted as time vs frequency RFI flag plot. Each dot represents a time-sample/freq-channel flagged as RFI affected by spec2fits.

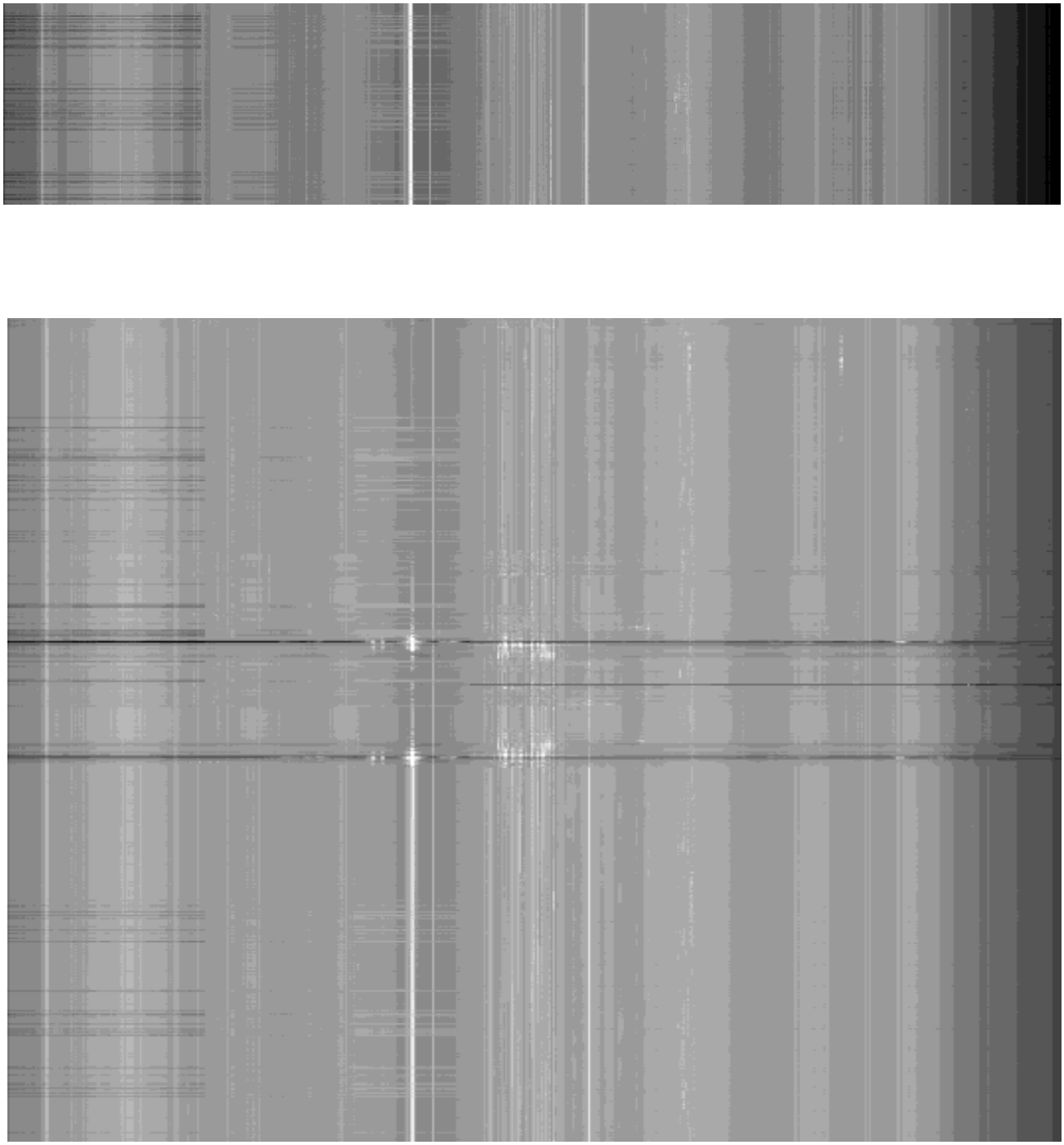

Figure 10: time (vertical direction) vs frequency (horizontal direction) waterfall image of the data in dbbc_201418911.fits (top panel), dbbc_201418912.fits (bottom panel). Time increases upwards, while frequency increases from left to right. The signal intensity is measured in dB and mapped in grey scale (white = strong, black = faint).

Figure 11: time (vertical direction) vs frequency (horizontal direction) waterfall image of the data in dbbc_201418913.fits (C band). Time increases upwards, while frequency increases from left to right. The signal intensity is measured in dB and mapped in grey scale.

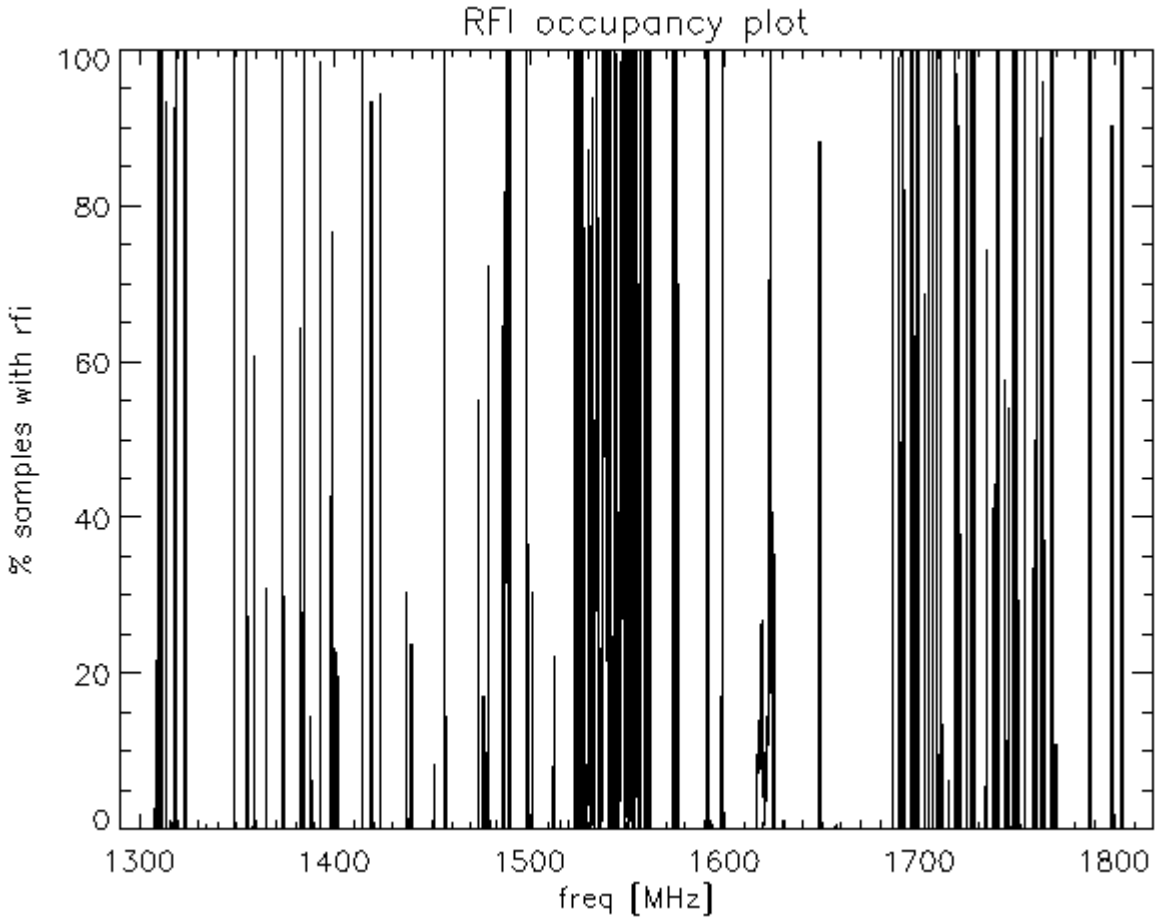

Figure 12: RFI occupancy plot of data in the FITS file dbbc_201418911.fits (L band).

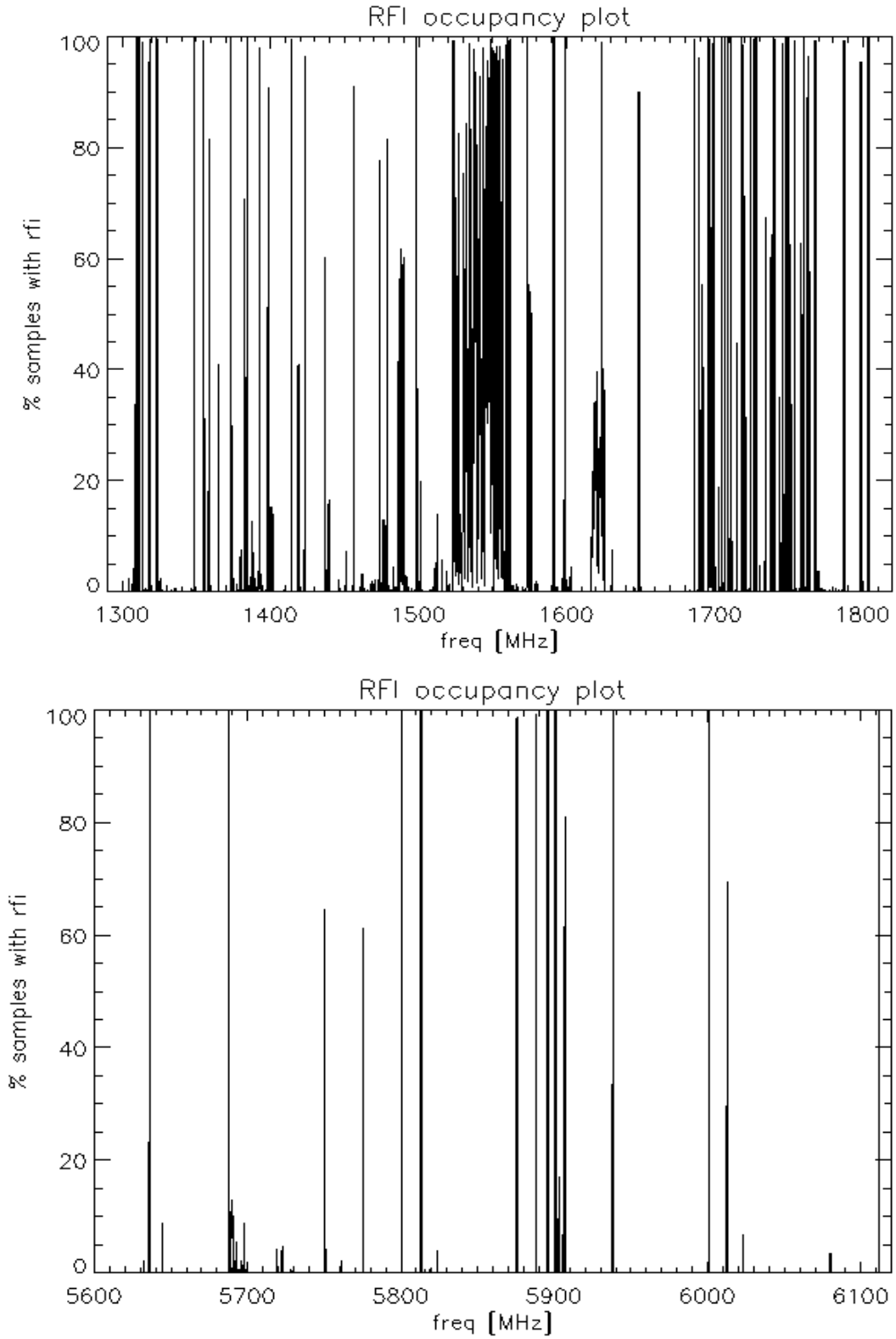

Figure 13: RFI occupancy plot of data in the FITS file dbbc_201418912.fits (L band, top panel) and dbbc_201418913.fits (C band, bottom panel).

5 Conclusions

In this report a monitoring system thought for RFI purposes has been described. At present it is installed and is working to the SRT, however its porting for whatever telescope can be done very easily on condition that a DBBC be available; of course, however, other back-ends can be used to calculate the spectra such as ROACH/ROACH2 boards.
Future developing works will be the creation of a “clever” queried database exploitable to avoid well known interference that can certainly ruin or decay the radioastronomical observations.

6 References

[1] Tuccari, G. “Dbbc – a wide band digital base band converter” in [Third General Meeting], Vandenberg, N.R. And Bayer, K.D., eds [2004]

[2] Comoretto, G., Melis, A., Tuccari, G., “A wideband multirate FFT spectrometer with highly uniform response,”, Experimental Astronomy 31, 59-68 (2011).

[3] Melis, A., Comoretto, G., “A 512 MHz Polyphase Filterbank with overlapping bands,” Arcetri Observatory Technical Reports 1-2011, INAF (2011)

[4] Orlati A., Buttu M., Melis A., Migoni C., Poppi S., Righini S., “The control software for the Sardinia Radio Telescope ,” Software and Cyberinfrastructure for Astronomy II, Proc. Of SPIE vol. 8451, 2012

[5] Taylor, A.R. & Salter, C.J., “GALFACTS: The G-ALFA Continuum Transit Survey”, in “*The Dynamical ISM: A celebration of the Canadian Galactic Plane Survey*”, ASP Conference Series, Vol. 438, page 402 (arXive:1008.4944)

[6] http://www.ira.inaf.it/~bartolini/rfi/dbbc2spec/data_format_considerations.html

[7] www.python.org

[8] *This research made use of Astropy, a community-developed core Python package for Astronomy (Astropy Collaboration, 2013)* www.astropy.org

[9] http://www.ira.inaf.it/~bartolini/rfi/dbbc2spec/dbbc2spec-1.0.tar.gz

Contents