RRI-EEG Internal Technical Report No. : 2
Date : 10 / 02 / 2021

Radio Frequency Interference Measurement at RRI Bangalore

K.B.Raghavendra Rao[1], R.Somashekar[1],

Raman Research Institute , Bangalore - 560080.

Abstract

The radio-frequency interference (RFI) measurements at lower frequencies from 30MHz to 430MHz carried out at the Raman Research Institute, the place which is located in Bangalore city environment. The purpose of conducting this study is to evolve the design of a low-frequency proto-type RF front end receiver system for the newly initiated RRI-SKA bridging activities at RRI, presently within the campus, till one identifies a better RFI free site. In this context, a simple RFI measurement setup is installed in RRI Laboratory terrace and measurements carried out, with an aim to choose a relatively clean band within this frequency range. The RF spectral data were collected and processed to identify RFI signals. In this document, the RFI data acquisition set-up and analysis along with the series of plots which describes the channel power of RFI, spectral occupancy etc., as a function of frequency are presented.

1. Intoduction

In Radio Astronomy experiments, the received power from an antenna which is pointed at the sky, may be partially contaminated with Radio Frequency Interference signals. It is very hard to distinguish between the astronomical radio signals and low level interference sources in the received power spectrum. RFI is classified into two types based on the effects that it produces in the spectrum of the received signal. Broadband RFI is a noise-like signal and it causes an increase in the power level of the entire spectrum, and is usually generated in high power transmission lines, automobile ignitions and switching of inductive loads etc. Narrowband RFI is the occurrence of an unwanted discrete frequency or a band of frequencies affecting only a narrow part of the receiver's spectrum. Narrowband RFI is often produced by communication broadcast transmitters for TV, short distance FM radio services, aircraft and satellites etc.,. Also, RFI enters the highly directive antennas of the radio telescope even through the side lobes.

However, it may also enter through the main lobe during observations at lower elevation angles. This additional power introduced by RFI can be seen as increase in noise temperature of the receiver. RFI can affect radio astronomy observations in several ways at its most extreme case of in-band strong manmade interference, saturates the receiver's pre-amplifiers which leads to the generation both harmonic and inter modulation distortions. RFI can also contaminate some frequency channels within an observation band while leaving other frequencies usable, rather it is difficult to filter out depends on how far it's position in the frequency axis. Low power RFI may only be detectable in long integrations of data where it is hard to excise.

2. Motivation

The primary motivation of the RFI measurement is to estimate the RFI signal power available along with the sky signal of the RF spectrum band in frequency range from 30MHz to 430MHz and the possible utilisation of the usable sub-band in this range. The total measured averaged RF power spectrum acquired over a long period of duration in each frequency channel determines the RF input power for the design and implementation of a low-noise front-end receiver system, aimed for low frequency SKA-RRI astronomical activities.

3. Measurements

In our requirement, the RF spectrum data acquisition and it's analysis in city environment were carried out to estimate the strength of the RFI signals in the frequency range 30MHz to 430MHz. An RFI (wire conical broad band antenna) antenna designed in-house and used for various field applications (Coutesy Dr.A.Raghunathan) at the Raman Research Institute has been incorporated at the EEG (Electronic Engineering group) laboratory terrace for the RFI spectrum acquisition. The signal is transported to the lab using a coaxial RF cable and connected to the Keysight Field fox spectrum analyser. The spectrum analyser was remotely controlled to acquire the trace data through Ethernet using dedicated commands which are compiled in a script developed using C program. The set-up diagram is as shown in figure 1.

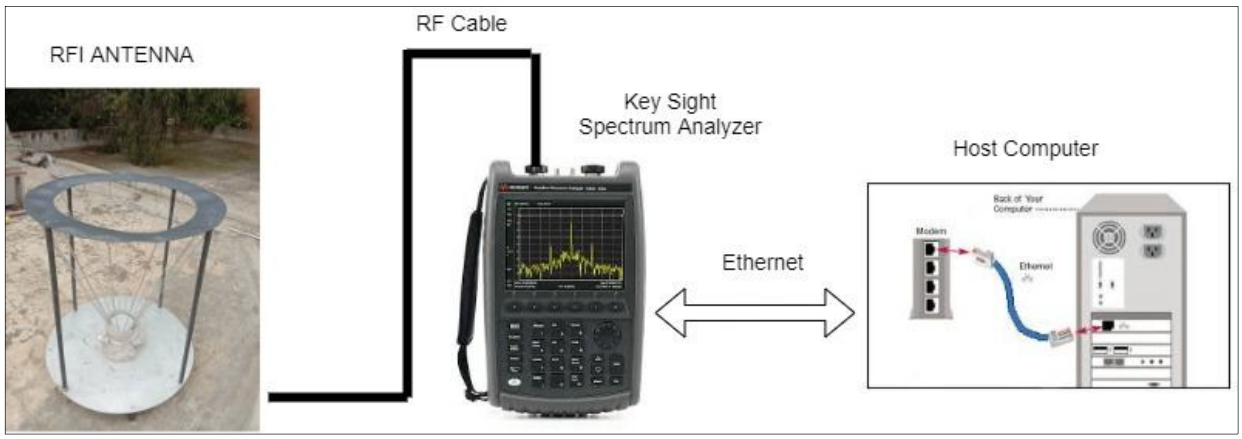

Figure 1 Radio Frequency Measurement Set-up diagram

The typical Spectrum Analyzer settings are as follows :

`Instrument ID:` **`Keysight Technologies,N9915A,MY53101604,A.08.18`**

System date: yyyy,mm,dd

Start frequency: 30 MHz

Stop frequency: 430 MHz

Center frequency: 230 MHz

Frequency span: 400 MHz

Resolution BW: 100 KHz

Video BW: 50 KHz

Sweep points: 1001

Sweep time: 2 Secs

Data form: ASC,0

Start time: hh,mm,ss

End time: hh,mm,ss

As a reference, the Return loss (S11) of the RFI antenna used for this experiment was measured before starting the data acquisition, and is as shown in figure 2. It is found that this antenna is suitable for the band we intend to carry out the RFI measurements.

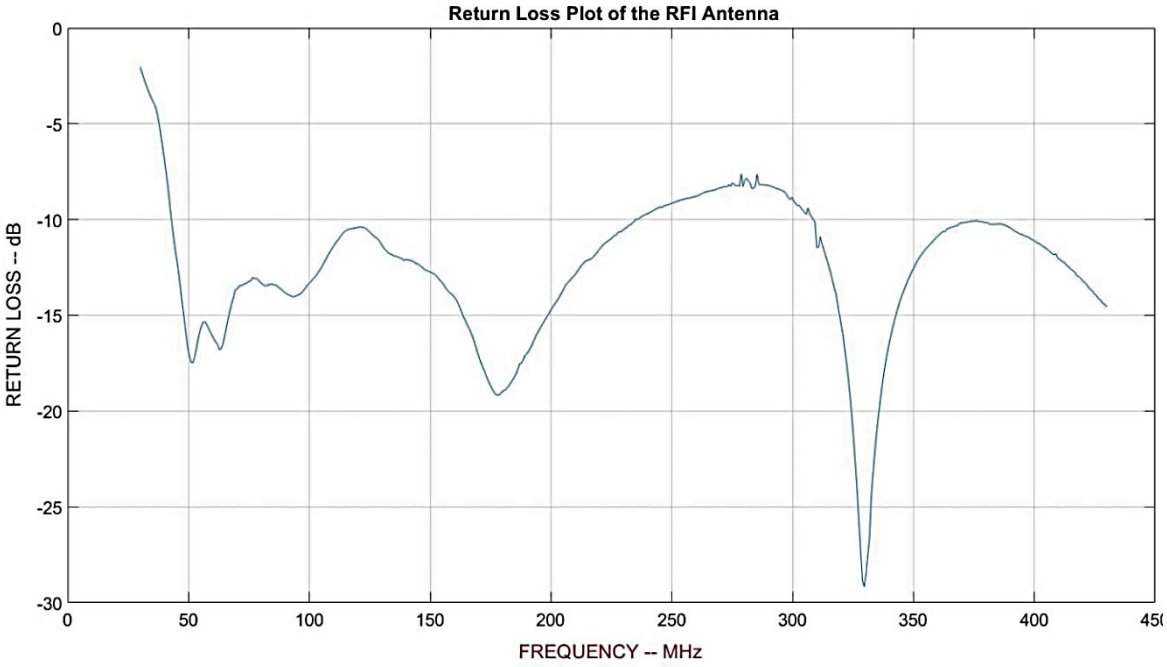

Figure 2. Frequency Vs Return Loss plot of the RFI antenna

3.1 Acquisition procedure :

The RFI data acquisition was carried out through an automated C-program in Linux operating system. The initial settings of the spectrum analyser like start frequency, stop frequency, amplitude reference level, internal attenuation, resolution band width (RBW), sweep time and number of points in each trace are set by issuing analyzer's dedicated commands through ethernet. Then all the frequency points of the corresponding trace are saved in the reference file in the host computer. The time taken for saving each trace is approximately about 2 seconds and with an user defined sleep time, we acquired the each trace data once in 6 to 10Secs depending on the observational requirements.

The RF spectrum data acquisition is done initially for one day in the frequency range between 30MHz to 430MHz and it is repeated for several hours continuously.

3.2 Analysis and Results :

The RFI analysis has been carried out using MATLAB software developed for this purpose. The matlab program produces the plots of basic averaged RF spectrums, minimum and maximum strengths of the signals and the total power of each scan in a given bandwidth setting as function of acquisition time.

The analysed plots of the data set acquired for 26 hrs duration on 17th August 2020 are as shown below.

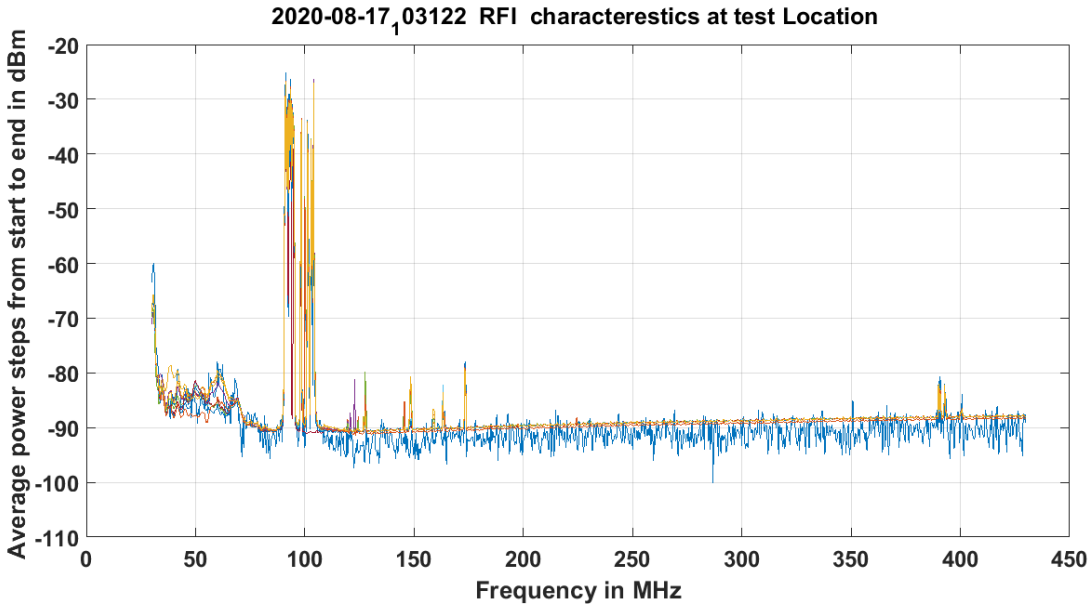

Figure 3a. Averaged RF spectrum

The above plot describes the basic acquired RF spectrums averaged for 500 averages. The signal to noise ratio of the signals in the RF spectrum is referenced to noise floor of the instrument which is limited to around -93 dBm with 100 KHz bandwidth. The FM band signals ranging from 91MHz to 107MHz are predominant in the frequency spectrum with very high amplitude levels greater than -30dBm. In the low frequency region between 30 to 70MHz, the maximum RFI signal levels are about -80dBm. Between 120MHz and 180MHz, the TV broadcast and other radar signals occupy the frequency spectrum with power levels up to-80dBm.

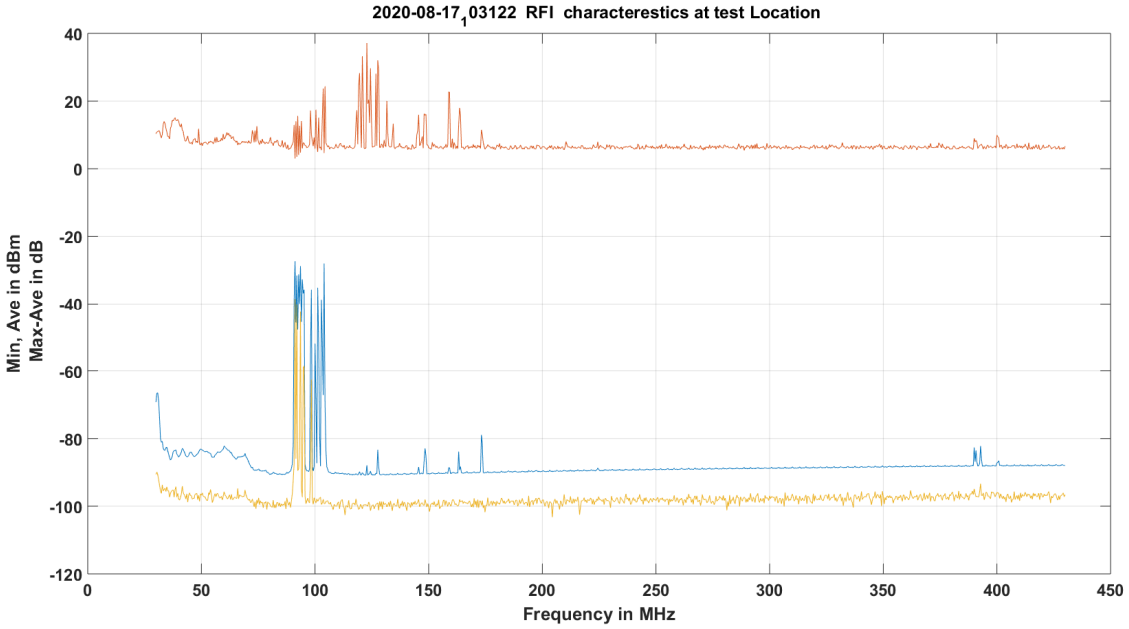

Figure 3b. Minimum, Average and Max-Average strengths of the RF spectrums in the acquired data

The minimum (yellow trace), Average (blue trace) and difference of maximum and average (brown trace) respectively are plotted as shown in figure 3b. The total power of the each frequency spectral scan within a specified bandwidth has been plotted as a function of acquired time interval and is shown in figure 3c. From the figure 3c, it can be observed that at any given time interval, the total RFI power is around -60dBm except for few intervals of time, when the FM channels are removed, and the observational band is limited from 120MHz to 420MHz. While in figure 3d, the water fall plot of the trace data, one can observe the spectral features of the entire band over the complete observation time. From the plots 3c and 3d one can nicely correlate that the total power level drop from 13hrs to 19hrs correspond to the transmission OFF of a few FM channels in the waterfall plot.

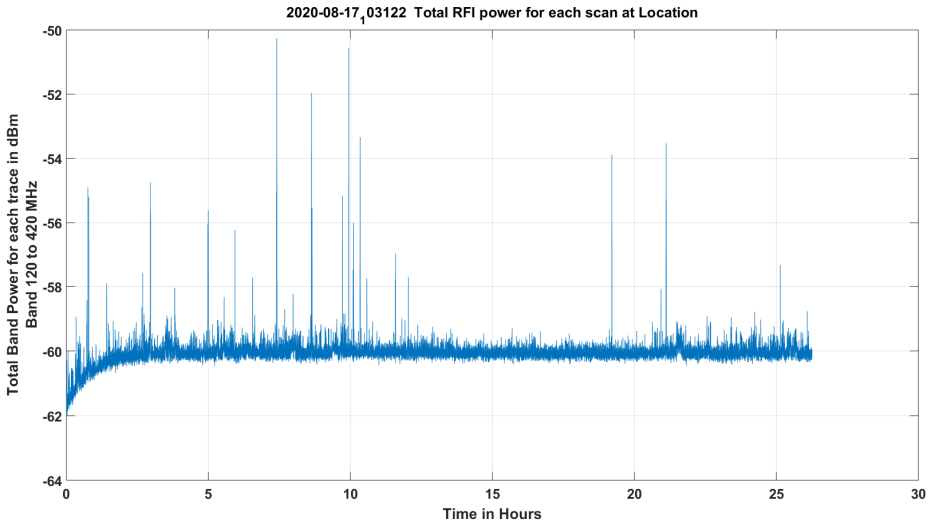

Figure 3c. Total RFI power of each scan within a specified bandwidth.

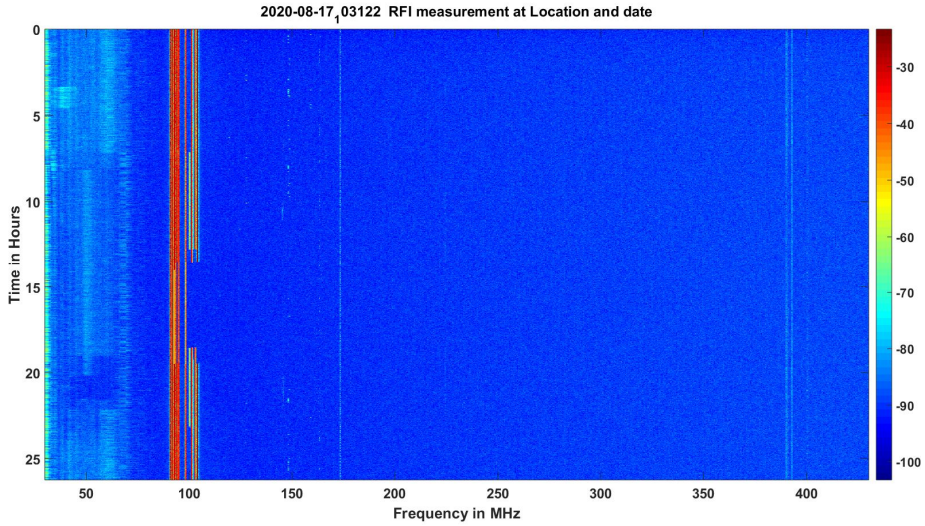

Figure 3d. Water fall plot showing the RFI signal strength in the frequency spectrum from 30MHz to 430MHz

In the above experimental setup, we notice that the spectrum analyser noise figure dominates our measurements and the low level RFI are buried in the noise. To better understand the RFI scenario in the band from 120MHz to 430 MHz, we have adopted the scheme as shown in figure 4 taking the measurements by notching the FM band and adding gain stages to have a better SNR.

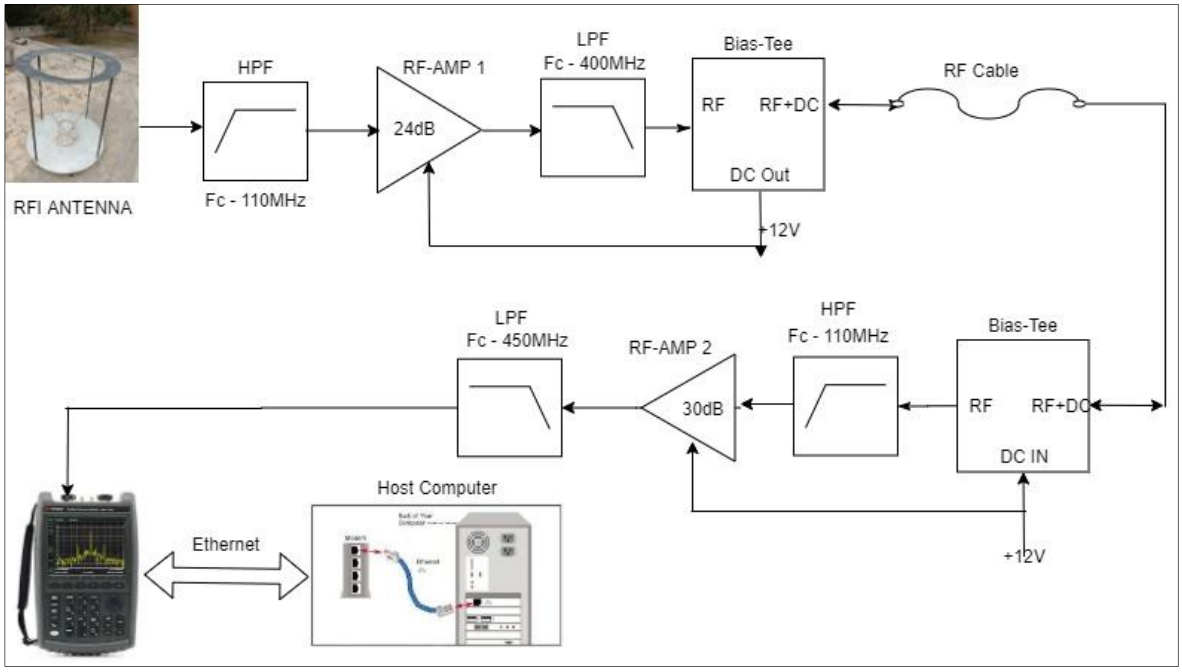

Figure 4. Modified set-up scheme to detect low strength RFI signals

To measure the lower level RFI signals below -90dBm noise floor of SA, a dual high-pass filter having -3dB cut-off at 120MHz which attenuates the FM signal band by around 70dB level along with a 30dB amplifier have been included at the RFI antenna output. Since FM band is rejected by a large extent of about 70dB, the signal to noise ratio of the RFI signals were elevated by gain and limited by the noise figure of the amplifier used and the data acquisition were carried out. Figure 5 shows the complete band including the FM rejection. As our interest lies in the choice of good band beyond 120MHz, we had carried out most analysis in this part of the band.

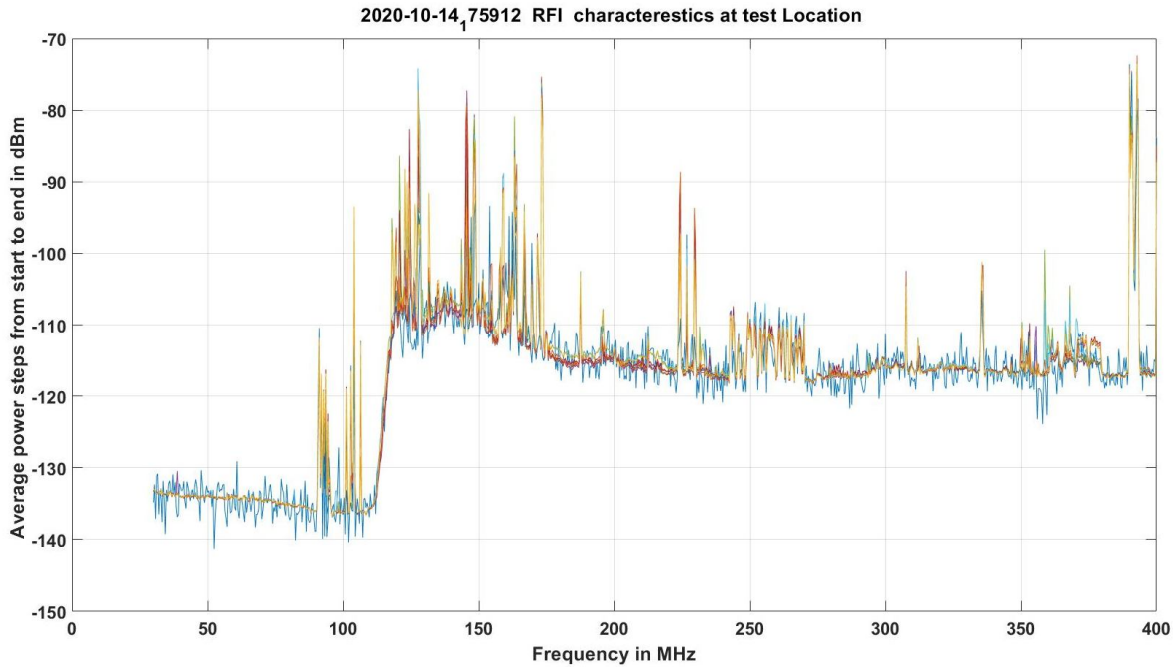

Figure 5a. Averaged RFI Spectrum in the frequency range 30MHz to 430MHz

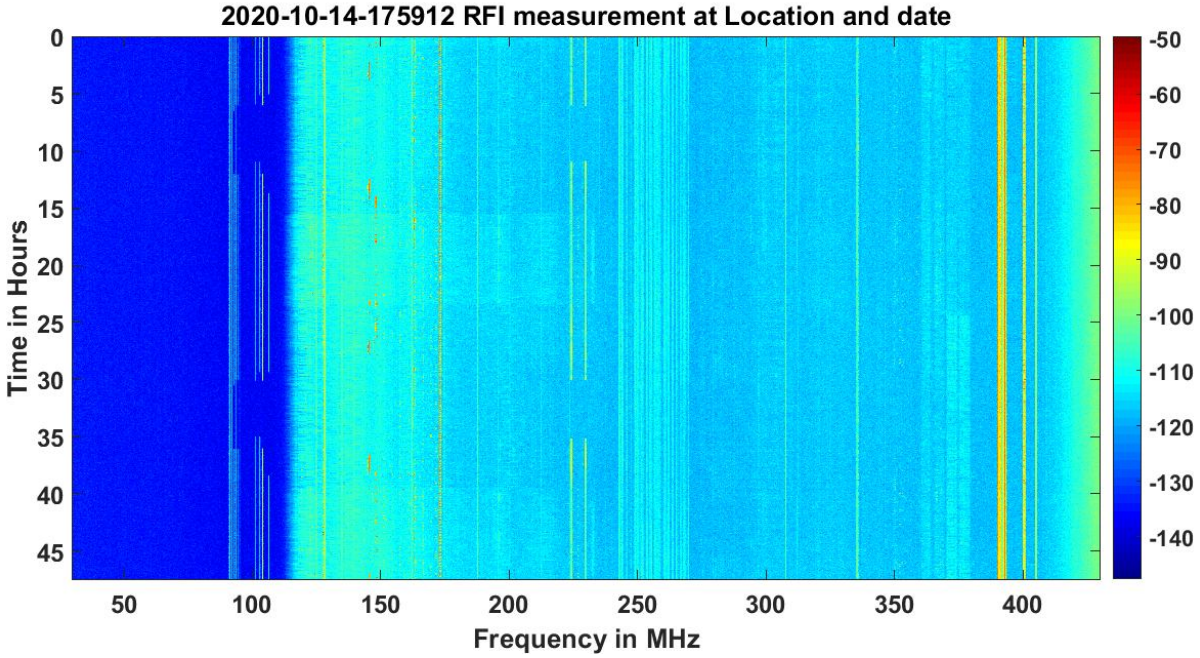

Figure 5b. Waterfall plot of each RFI spectrum over acquisition interval

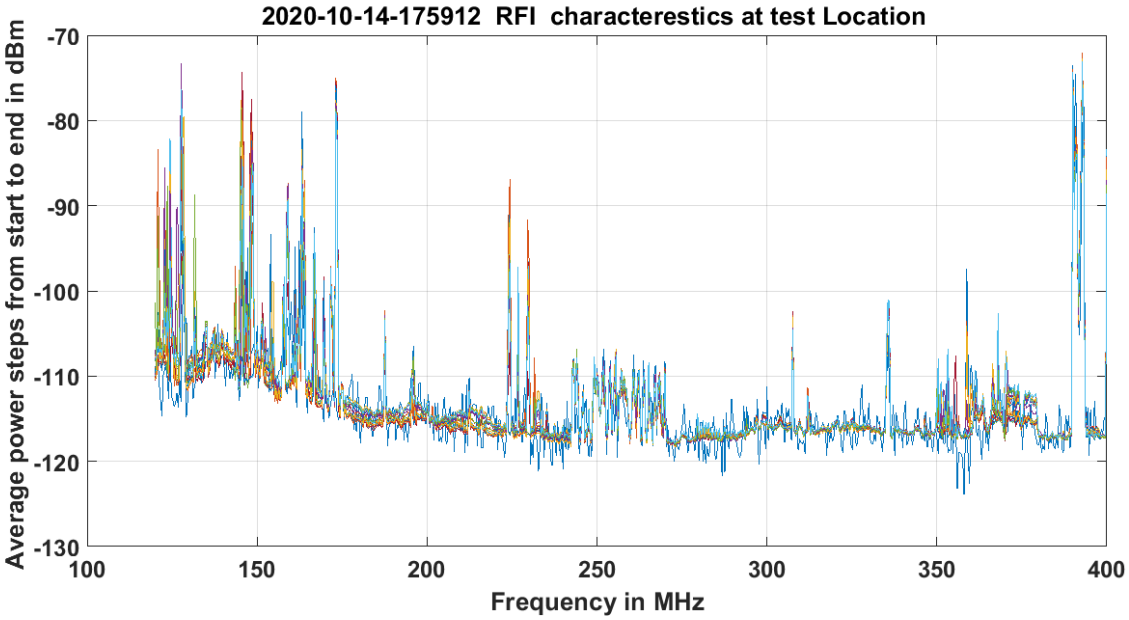

Figure 5c. Averaged RFI Spectrum in the frequency range 30MHz to 430MHz

The analysed plots of the RFI data with the modified scheme are as shown in the figure 5.

It can be observed from the above figure that we are able to achieve a lower noise floor and thus pick up RFI as well as sky above -115 dBm It can be realized from the above figure 5a that, the noise floor has been reduced to an average level of around -115dBm and all the lower level RFI can be seen for the same 100 KHz RBW. The overall gain/loss of the RF chain has been taken into account in the analysis program. We have also calibrated the system and applied the same in the plots to ensure that the system Gain/Losses are taken care and what is seen is the actual signals received through the antenna at 100KHz resolution bandwidth.

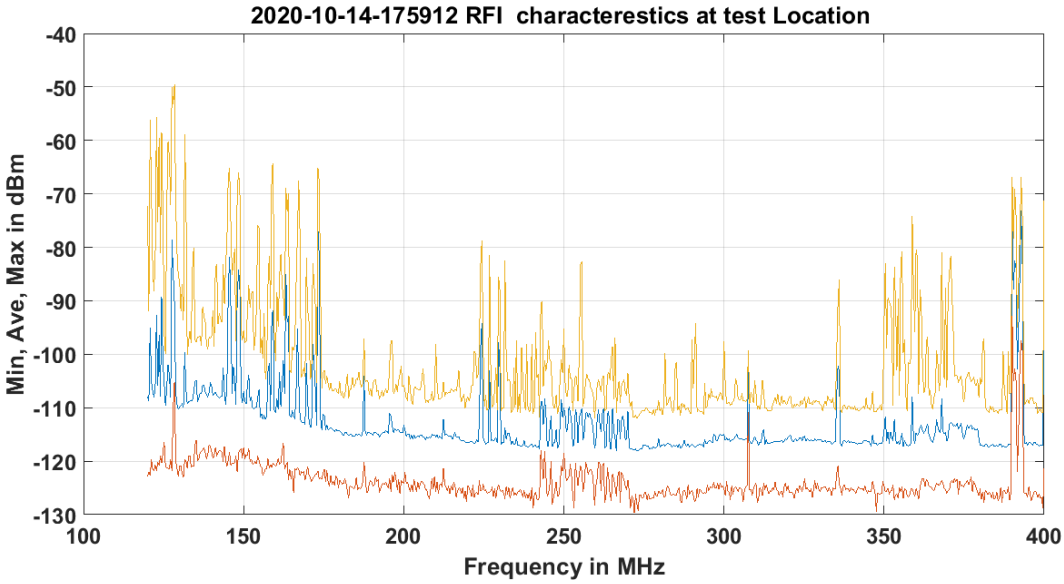

Figure 5d. Minimum and Maximum strengths of RFI

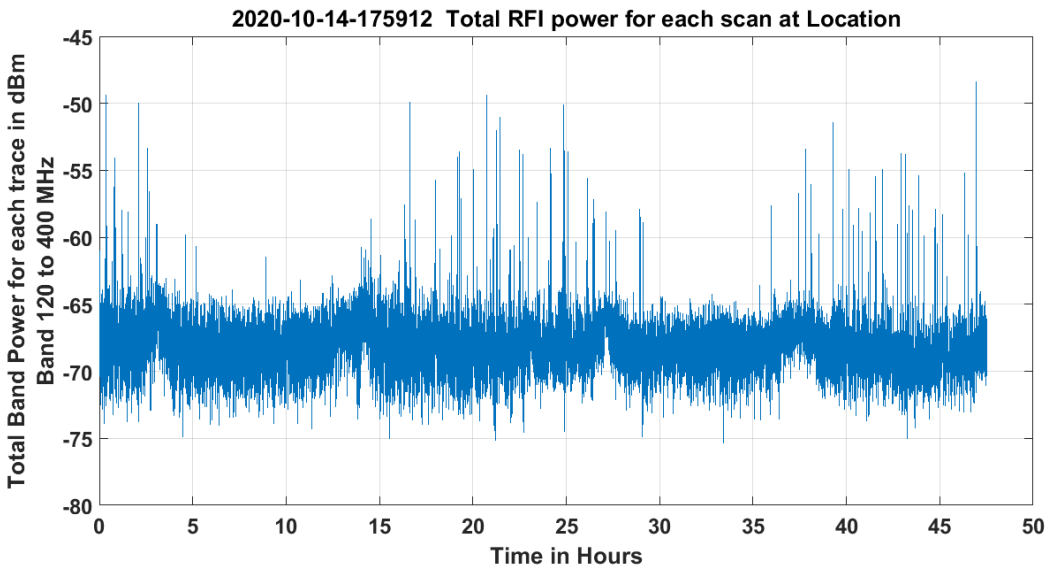

Figure 5e. Total Power plot of each RFI spectrum over the duration of acquisition interval

From the plot 5b, it can be viewed that the average RFI power level is -80dBm. A few probable RF strong signals in the RF spectrum are Bangalore TV 175MHz and intelsat 230MHz. From the figure 5c, it's observed that the typical power of RFI signals within the acquired RF spectrum at any given instance is around -80dBm, and it's occasionally peaking up to -70dBm for a short duration of time.

RFI Occupancy Analysis

The RFI occupancy analysis has been carried out to understand the duration of RFI presence at various threshold above noise floor. A frequency channel is occupied by RFI, as long as the measured level is above certain threshold from the noise level. The RFI spectral occupancy analysis for the acquired RF spectral data are done for different threshold levels from the calibrated noise floor. The corresponding percentage of occupancy are plotted in the frequency range from 120MHz to 400MHz as shown in the figures from 6a through 6h, in percentage against different threshold. From the above RFI occupancy plots, it is observed that the RFI frequencies 175MHz and 380MHz are consistently with 100% occupancy of the acquired duration in the RF spectrum. These frequencies can be avoided permanently by choosing reasonable form factor band pass filters.

Threshold levels 15dB and above, the average percentage of RFI occupancy over the full acquired time keeps diminishing. Also the RFI occupancy about 20dB level, we almost have a

clean usable band from 180MHz to 380MHz. Hence, with good system dynamic range and a head room for RFI power upto 20dB system can be designed to work in this sub-band. With this threshold, which corresponds to a loss of 3 bits of ADC, sky observations are possible with reasonable linearity in the high gain good dynamic front end receiver system.

The RFI occupancy for different thresholds plotted for the frequency range from 120MHz to 400MHz is shown in the figures 6a to 6h.

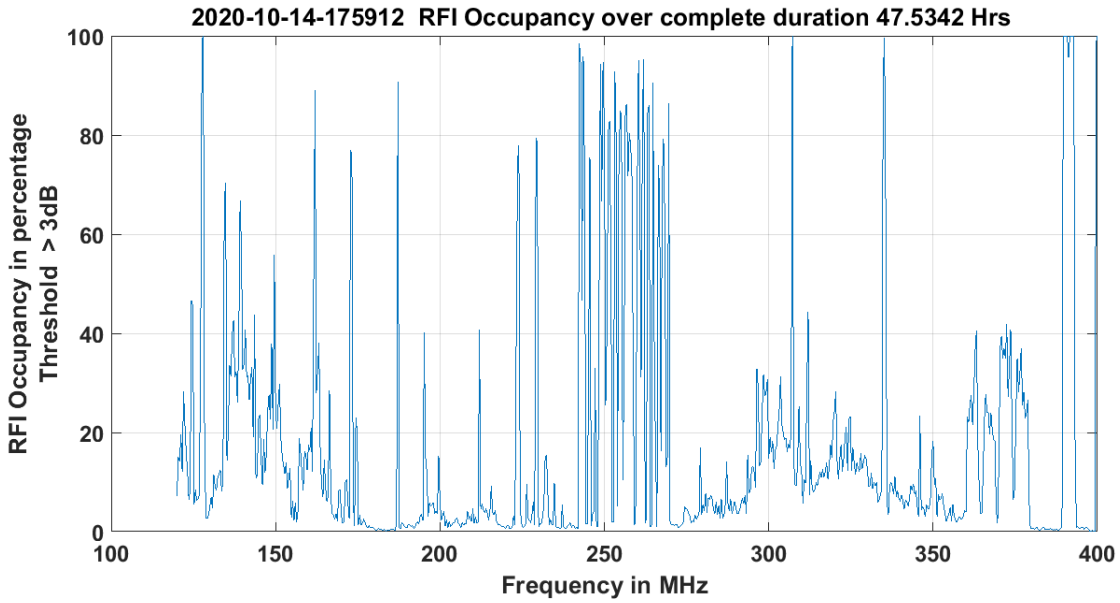

Figure 6a. RFI spectral occupancy above 3dB threshold level

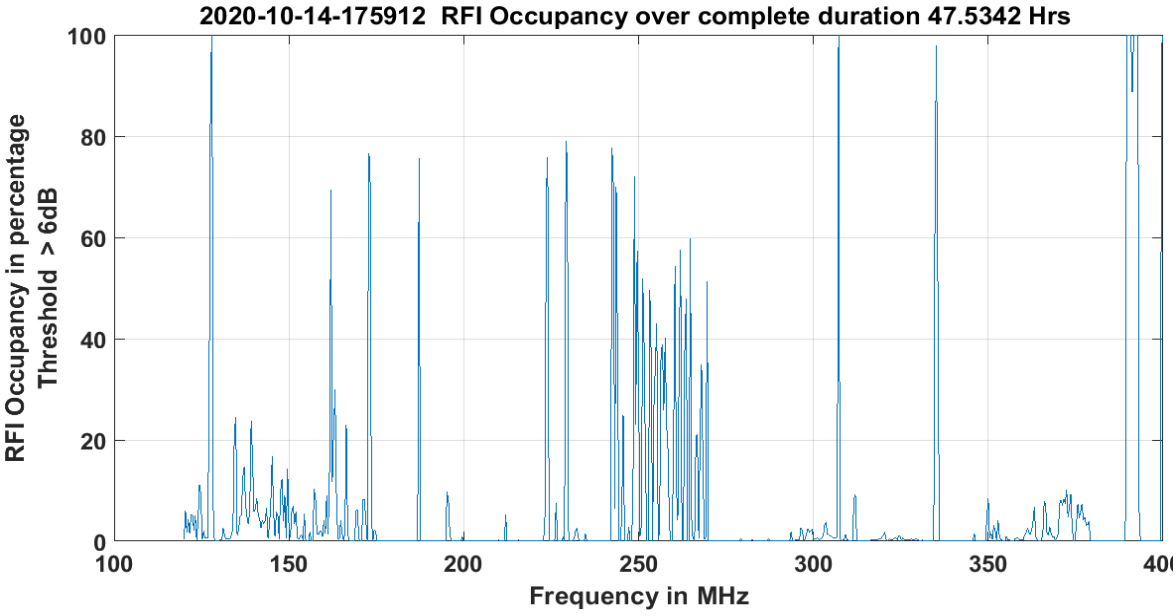

Figure 6b. RFI spectral occupancy above 6dB threshold level

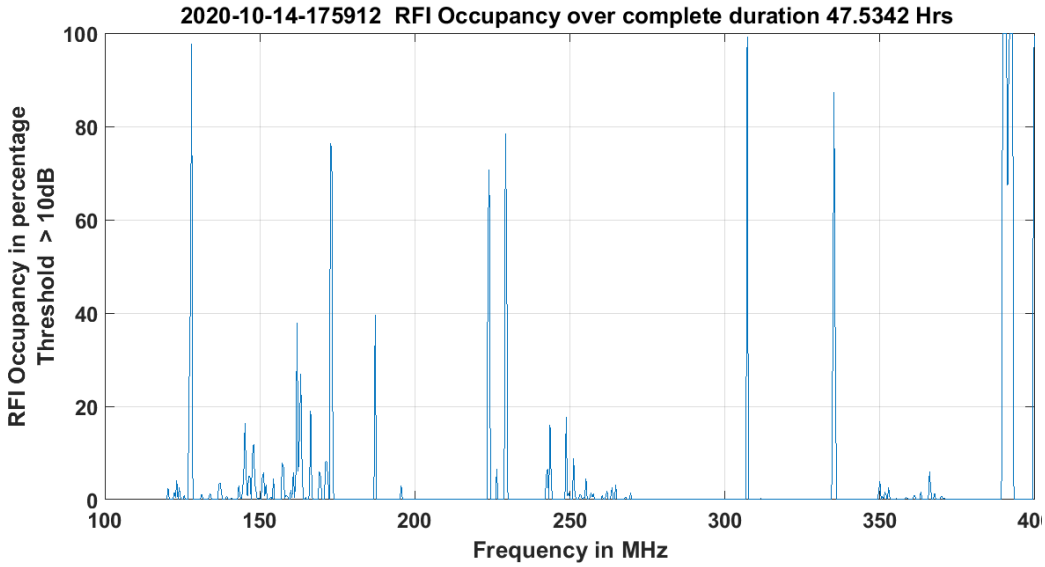

Figure 6c. RFI spectral occupancy above 10dB threshold level

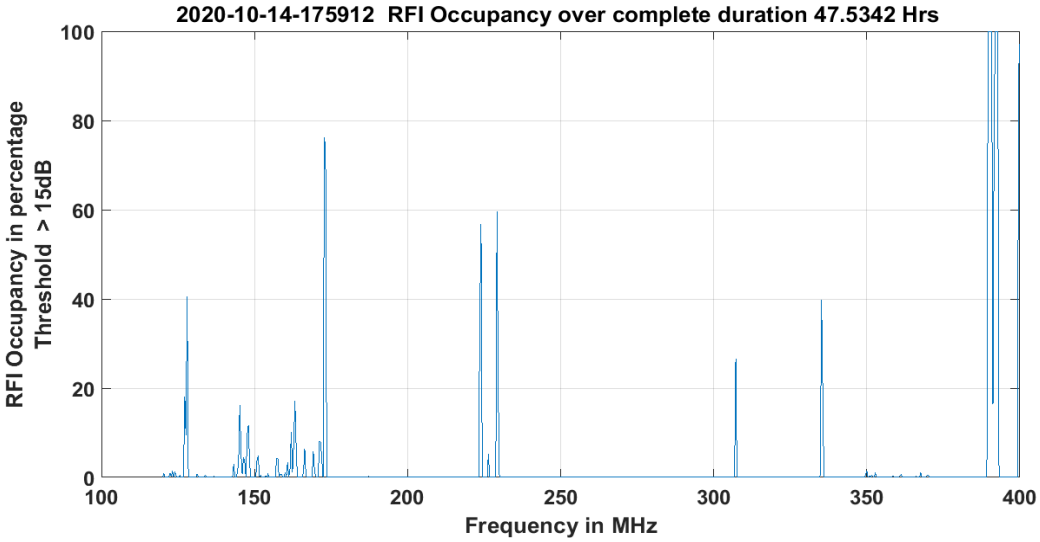

Figure 6d. RFI spectral occupancy above 15dB threshold level

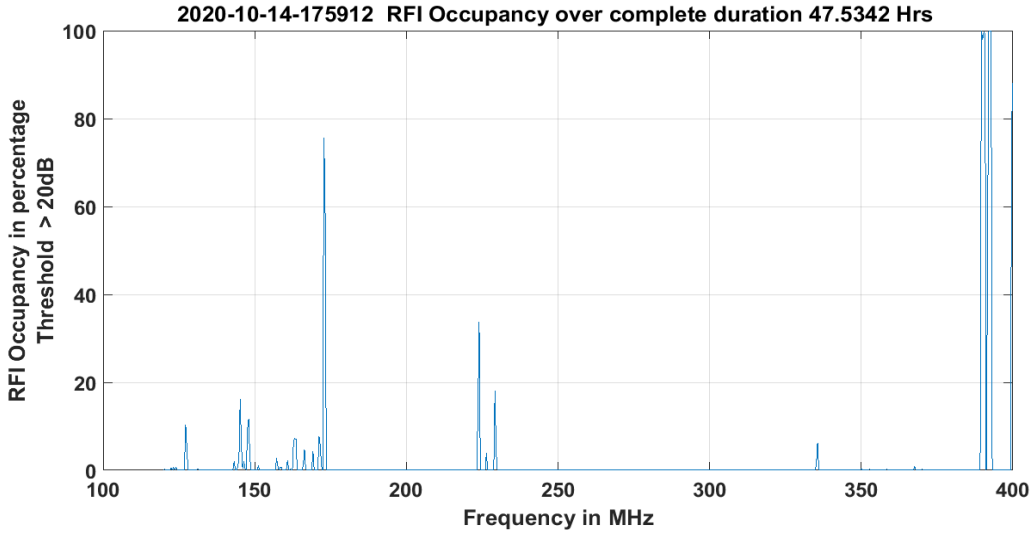

Figure 6e. RFI spectral occupancy above 20dB threshold level

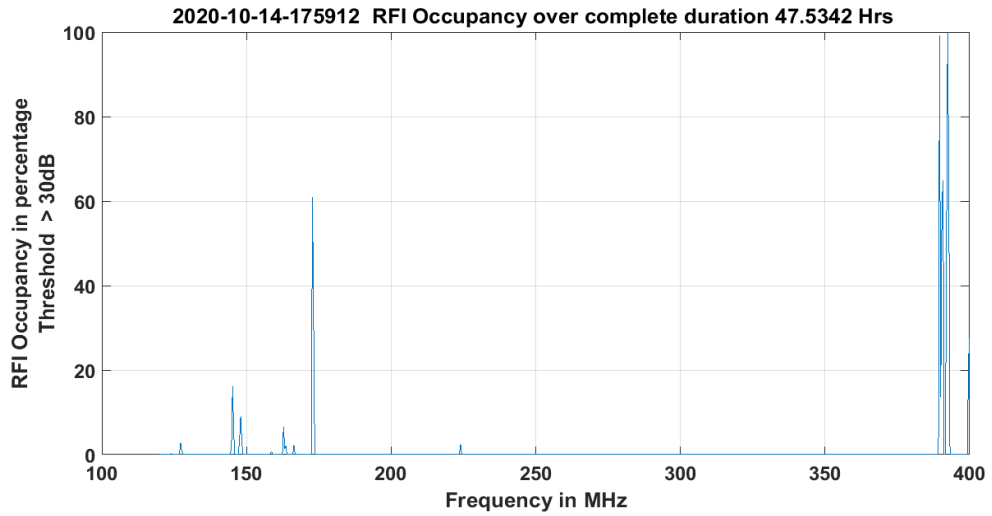

Figure 6f. RFI spectral occupancy above 30dB threshold level

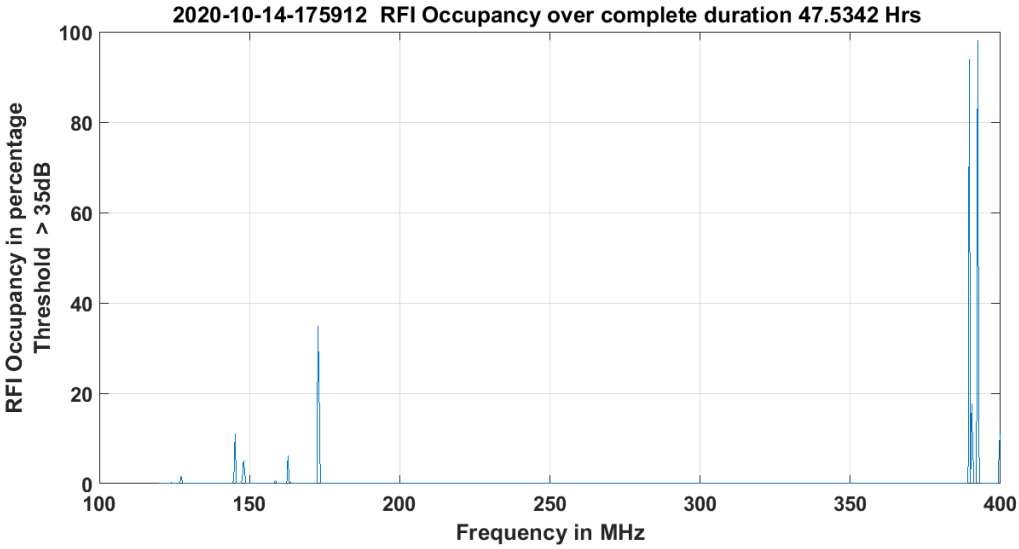

Figure 6g. RFI spectral occupancy above 35dB threshold level

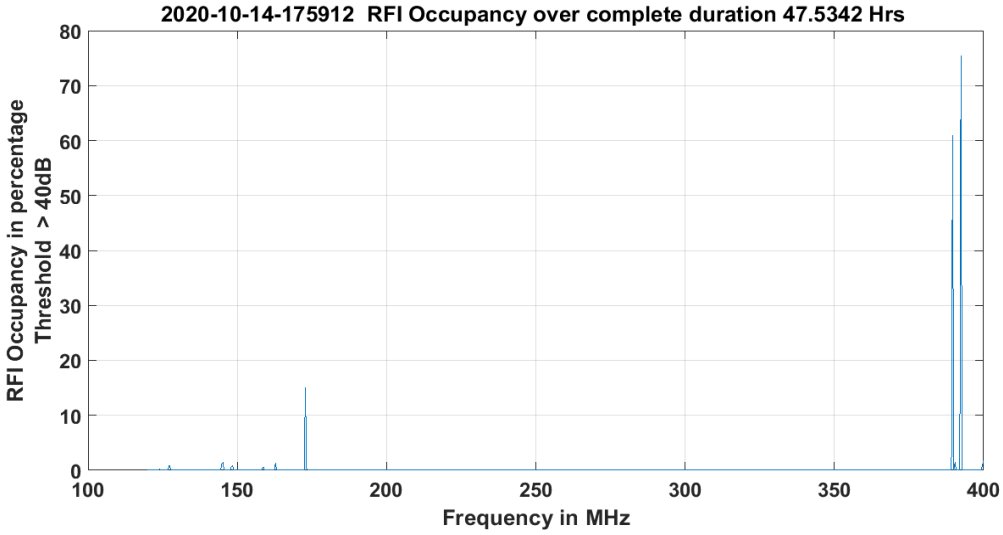

Figure 6h. RFI spectral occupancy above 40dB threshold level

4. Conclusion :

The RFI data acquisition were carried out in the frequency range from 30MHz to 430MHz using RFI antenna along with the necessary RF circuitry. A Keysight spectrum analyzer were used to acquire the RF spectrum for several hours and days. A systematic instructions and commands were issued to the spectrum analyzer using linux based C program through Ethernet. The acquired data were analyzed using Matlab program. From the analysis it's concluded that the preliminary data shows strong 90-110MHz FM band and it's power is -20dBm. An attenuation of greater than 40dB is required to reject FM band for radio astronomical observations. The average RFI total power within the frequency range 120MHz to 400MHz is typically -60dBm. To measure lower level RFI below -90dBm, the receiver was modified and acquired for longer durations in the frequency range from 120MHz to 400MHz. The average RFI power in this frequency range is about -85dBm. The frequencies 175MHz and 380MHz are consistently present at all the times. The RFI spectral occupancy has been evaluated by choosing thresholds from 3dB to 40dB are plotted. From the occupancy plots, it can be inferred that system can be designed for carrying out sky observations in the band from 180MHz to 380MHz with an headroom of 20dB for RFI. This can be ensured by choosing good roll off low-pass and high-pass filters by filtering the FM band, 175MHz and 380MHz frequencies. Hence, we propose to work with this numbers for a possible low frequency receiver system design for use in the campus, may be nearly an octave band from 180MHz to 350MHz.

Acknowledgements

We thank Prabu for the thrust and initiative to conduct the RFI measurements in the campus and the constructive discussions at every stage during this work. We thank Santhosh who had developed the linux based C- language code for the spectrum analyzer data acquisition for EOR/ERA projects, which was adapted with slight modification to suit our requirement.

We wish to express our sincere thanks to A. Raghunathan, for lending and supporting us in setting up the low frequency antenna for the RFI measurements. We would also like to thank Vinod for joining and helping us in setting up the antenna in EEG roof top.

We also wish to express our sincere thanks to all EEG members for involvement in discussions and providing valuable ideas at all stages during this work.

ANNEXURE A

RFI Data acquisition C program

```
/* RFI_MEAS - Acquire trace from spectrum analyzer through LAN (TCP)
*/
/* Santosh */
/* Modification by R.Somashekar for adapting to the RFI requirements
```

```
RFI_MEAS creates a socket, establishes TCP connection using port 5025
and given static IP address. It then creates a header file to store
the values of all important parameters such as start and stop
frequency, sweep time and points, etc. Later it initiates a single
trigger, collects the trace and stores it in a .DAT file. The number
of traces to sweep has to be given as an argument while executing the
program.
*/
/*
c************************************************************
c History
c
c Santosh 12Nov14 original version
c Santosh 17Nov14 modified for multiple trace retrieval
c Somashekar July 2015 modified for optimizing Sweep, BW and Trace
data
c Somashekar 2016 Modified to work remotely with Ethernet
c Somashekar and Raghavendra for introducing time delays and
c  optimising
c  ************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <bits/socket.h>
#include <string.h>
#include <sys/select.h>
#include <sys/time.h>
#include <errno.h>
#include <netinet/tcp.h>
#include <netinet/in.h>
#include <time.h>

#define SOCKETS_IP_ADDRESS            "192.168.1.97"
#define SOCKETS_PORT                  5025
#define SOCKETS_BUFFER_SIZE           1024
#define SOCKETS_TIMEOUT                    10

int WriteString(int MySocket,char string[])
{
      int retval;

      if((retval=send(MySocket,string,strlen(string),0))==-1) {
            printf("Error: Unable to send message (%i)...\n",errno);
            perror("sockets"); // Print error message based on errno
```

```
            exit(1);
      }
      return retval;
}

int WaitForData(int MySocket)
{
      fd_set MyFDSet;
      struct timeval tv;
      int retval;

      // Wait for data to become available
      FD_ZERO(&MyFDSet);               // Initialize fd_set structure
      FD_SET(MySocket,&MyFDSet);       // Add socket to "watch list"
      tv.tv_sec=SOCKETS_TIMEOUT; tv.tv_usec=0; // Set timeout
      retval=select(MySocket+1,&MyFDSet,NULL,NULL,&tv); // Wait for
change

      // Interpret return value
      if(retval==-1) {
            printf("Error: Problem with select (%i)...\n",errno);
            perror("sockets"); // Print error message based on errno
            exit(1);
      }
      return retval; // 0 = timeout, 1 = socket status has changed
}

int ReadString(int MySocket,char *buffer)
{
      size_t actual;

      // Wait for data to become available
      if(WaitForData(MySocket)==0)
      {
            // Timeout
            printf("Warning: Timeout...\n");
            return 0;
      }

      // Read data
      if((actual=recv(MySocket,buffer,100,0))==-1) {
            printf("Error: Unable to receive data (%i)...\n",errno);
            perror("sockets"); // Print error message based on errno
            exit(1);
      }
      else {
      //printf("Received data size: %zu\n",strlen(buffer));
            buffer[actual]=0;
      }
      return actual;
}

void ReadTrace(int MySocket,char *outfile)
{
      int actual,j,i,lcount,pkt_size,flag;
      char *ctoken,*nul = "(null)";
```

```
	char trace_data[8192];
	char buffer[10];
	FILE *fp;
	size_t length;

	fp = fopen("trace.dat","a");
	while(pkt_size < 16016) // Change the iteration condition
according to the sweep points
	{
		memset(trace_data, 0,sizeof(trace_data));  //clear the
variable
		flag = 0;

		// Read data
		actual=recv(MySocket,trace_data,sizeof(trace_data),0);
		trace_data[actual]=0;

		ctoken = strtok(trace_data,",");
		fprintf(fp,"%s\n",ctoken);
		while((ctoken != NULL))
		{
			lcount++;
			ctoken = strtok(NULL,",");

		 if(ctoken == NULL)
				;
		else if(((length = strlen(ctoken)) < 15) && (flag == 0))
			{
				fprintf(fp,"%s",ctoken);
				flag = 1;
			}
		else if(((length = strlen(ctoken)) < 15) && (flag == 1))
			{
				fprintf(fp,"%s\n",ctoken);
				flag = 0;
			}
			else
				fprintf(fp,"%s\n",ctoken);
		}
		pkt_size += actual;
	}
	pkt_size=0;
	fclose(fp);

}

void SetNODELAY(int MySocket)
{
	int StateNODELAY = 1; // Turn NODELAY on
	int ret;

	ret=setsockopt(MySocket,    // Handle to socket connection
		IPPROTO_TCP,          // Protocol level (TCP)
		TCP_NODELAY,          // Option on this level (NODELAY)
		(void *)&StateNODELAY,      // Pointer to option variable
		sizeof StateNODELAY);       // Size of option variable
```

```
	if(ret==-1) {
		printf("Error: Unable to set NODELAY option
(%i)...\n",errno);
		perror("sockets"); // Print error message based on errno
		exit(1);
	}
	return;
}

int main(int argc,char **argv)
{
	int MySocket,MyControlPort;
	struct hostent *hp;
	char SocketsBuffer[SOCKETS_BUFFER_SIZE];
	char *trace_buf;
	static char *ctoken;
	char modeopt[30];
	struct sockaddr_in MyAddress,MyControlAddress;
	unsigned int ControlPort;
	int status,lcount,i,retval,tcount=0;
	FILE *fh;
	clock_t begin, end;
	double time_spent;

	const struct tm *tm ;
	struct tm *cur_time_struct;
	long cur_time_long;
	int time_info1[6];
	char outfile[70],outhdr[70];

	// Wait for data to become available
	if(WaitForData(MySocket)==0)
	{
		// Timeout
		printf("Warning: Timeout...\n");
	}

	time(&cur_time_long);
	cur_time_struct = (struct tm *) localtime(&cur_time_long);
	time_info1[0]=cur_time_struct->tm_hour;
	time_info1[1]=cur_time_struct->tm_min;
	time_info1[2]=cur_time_struct->tm_sec;
	time_info1[3]=cur_time_struct->tm_mday;
	time_info1[4]=cur_time_struct->tm_mon+1;
	time_info1[5]=cur_time_struct->tm_year + 1900;
	sprintf(outfile,"%04d-%02d-%02d_%02d%02d%02d.dat%c",
		time_info1[5],time_info1[4],time_info1[3],
		time_info1[0],time_info1[1],time_info1[2],'\0');
	sprintf(outhdr,"%04d-%02d-%02d_%02d%02d%02d_hdr%c",
		time_info1[5],time_info1[4],time_info1[3],
		time_info1[0],time_info1[1],time_info1[2],'\0');

	// Create socket (allocate resources)
	if((MySocket=socket(
		PF_INET, // IPv4
```

```
            SOCK_STREAM, // TCP
            0))==-1) {
            printf("Error: Unable to create socket (%i)...\n",errno);
            perror("sockets"); // Print error message based on errno
            exit(1);
      }

      // Establish TCP connection
      memset(&MyAddress,0,sizeof(struct sockaddr_in));
      // Set structure to zero
      MyAddress.sin_family=PF_INET; // IPv4
      MyAddress.sin_port = htons(SOCKETS_PORT);
      // Port number (in network order)
      MyAddress.sin_addr.s_addr = inet_addr(SOCKETS_IP_ADDRESS);
      // IP address (in network order)

      if(connect(MySocket,(struct sockaddr *)&MyAddress,sizeof(struct
sockaddr_in))==-1)
      {
            printf("Error: Unable to establish connection to socket
(%i)...\n",errno);
            perror("sockets"); // Print error message based on errno
            exit(1);
      }
      // Minimize latency by setting TCP_NODELAY option
      SetNODELAY(MySocket);

      // Clear status and reset instrument to user settings

      WriteString(MySocket,"INST 'SA';*OPC?\n");
      ReadString(MySocket,SocketsBuffer);

      WriteString(MySocket,"FREQ:STAR 30E6;*OPC?\n");
      ReadString(MySocket,SocketsBuffer);

      WriteString(MySocket,"FREQ:STOP 250E6;*OPC?\n");
      ReadString(MySocket,SocketsBuffer);

      WriteString(MySocket,":SENS:SWE:POIN 1001;:SENS:SWE:TYPE
FFT;*OPC?\n");
      ReadString(MySocket,SocketsBuffer);

      WriteString(MySocket,":SENS:BAND:AUTO 0;:SENS:BAND
200E3;*OPC?\n");
      ReadString(MySocket,SocketsBuffer);

      WriteString(MySocket,":SENS:BAND:VID:AUTO 0;:SENS:BAND:VID
300E3;*OPC?\n");
      ReadString(MySocket,SocketsBuffer);

      WriteString(MySocket,":SENS:POW:GAIN 1;:SENS:POW:ATT:AUTO
0;:SENS:POW:ATT 0;*OPC?\n");
      ReadString(MySocket,SocketsBuffer);

      WriteString(MySocket,"DISP:WIND:TRAC1:Y:RLEV 0;*OPC?\n");
      ReadString(MySocket,SocketsBuffer);
```

```
WriteString(MySocket,"AMPL:SCAL LOG;*OPC?\n");
ReadString(MySocket,SocketsBuffer);

printf("User settings set\n");

fh=fopen("trace_hdr","w");

// Get instrument's ID string
WriteString(MySocket,"*IDN?\n");
ReadString(MySocket,SocketsBuffer);
fprintf(fh,"Instrument ID: %s",SocketsBuffer);

// Get system date
WriteString(MySocket,"SYST:DATE?\n");
ReadString(MySocket,SocketsBuffer);
fprintf(fh,"System date: %s",SocketsBuffer);

// Get start and stop frequency
WriteString(MySocket,":SENS:FREQ:STAR?\n");
ReadString(MySocket,SocketsBuffer);
fprintf(fh,"Start frequency: %s",SocketsBuffer);
WriteString(MySocket,":SENS:FREQ:STOP?\n");
ReadString(MySocket,SocketsBuffer);
fprintf(fh,"Stop frequency: %s",SocketsBuffer);

// Get center frequency
WriteString(MySocket,":SENS:FREQ:CENT?\n");
ReadString(MySocket,SocketsBuffer);
fprintf(fh,"Center frequency: %s",SocketsBuffer);

// Get frequency span
WriteString(MySocket,":SENS:FREQ:SPAN?\n");
ReadString(MySocket,SocketsBuffer);
fprintf(fh,"Frequency span: %s",SocketsBuffer);

// Get Res BW and Video BW values
WriteString(MySocket,":SENS:BAND:RES?\n");
ReadString(MySocket,SocketsBuffer);
fprintf(fh,"Resolution BW: %s",SocketsBuffer);
WriteString(MySocket,":SENS:BAND:VID?\n");
ReadString(MySocket,SocketsBuffer);
fprintf(fh,"Video BW: %s",SocketsBuffer);

// Get the no. of sweep points
WriteString(MySocket,":SENS:SWE:POIN?\n");
ReadString(MySocket,SocketsBuffer);
fprintf(fh,"Sweep points: %s",SocketsBuffer);

// Get sweep time
WriteString(MySocket,":SENS:SWE:ACQ?\n");
ReadString(MySocket,SocketsBuffer);
fprintf(fh,"Sweep time: %s",SocketsBuffer);

// Get data format
WriteString(MySocket,"FORM:DATA?\n");
```

```
	ReadString(MySocket,SocketsBuffer);
	fprintf(fh,"Data form: %s",SocketsBuffer);

	// Get system time before trace start
	WriteString(MySocket,"SYST:TIME?\n");
	ReadString(MySocket,SocketsBuffer);
	fprintf(fh,"Start time: %s",SocketsBuffer);

	// Initiate single sweep
	WriteString(MySocket,"INIT:CONT 0;*OPC?\n");
	ReadString(MySocket,SocketsBuffer);
	printf("Trigger mode set? (1-Yes 0-No): %s",SocketsBuffer);

	for(i=0;i<atoi(argv[1]);i++)
	{
		/*if(i>0){
		end = clock();
		time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
		printf("Time between sweep trigger: %f\n",time_spent);
		}*/
		// Collect trace data

		//begin = clock();
		WriteString(MySocket,"INIT:IMM;*OPC?\n");
		ReadString(MySocket,SocketsBuffer);
		if( SocketsBuffer )
		printf("Single sweep triggered?: Yes\t Trace
%d\n",++tcount);

		//usleep(4000000);
		WriteString(MySocket,"TRACE:DATA?\n");
		ReadTrace(MySocket,outfile);
		//end = clock();
		//time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
		//printf("Time between trigger initiation and end of trace
retrieval: %f\n",time_spent);
	}

	// Get system time after trace ends
	WriteString(MySocket,"SYST:TIME?\n");
	ReadString(MySocket,SocketsBuffer);
	fprintf(fh,"End time: %s\n",SocketsBuffer);

SocketMainClose:

	// Close main port
	if(close(MySocket)==-1)
	{
		printf("Error: Unable to close socket (%i)...\n",errno);
		perror("sockets"); // Print error message based on errno
		retval=1;;
	}

	exit(retval);
}
```

==========

ANNEXURE - B

RFI data Analysis MATLAB program

```
% Spectrum Analyser analysis code -- Trace , Average  and
  Water Fall plots
% R.Somashekar 13 July 2020

% First  dummy and second column of time axis in each row is taken and
removed
% RFI_P3 added file name and for extracting time info.
% Next option is to remove linear in Real Imginary format.
% Since SA No real and imaginary. Data is log or linear units.

% To be done: 14July2020
% To read the header file and take the information of
% Start Date and time, Start and Stop Freq. RBW,
% External Fixed gain or measured gain over the same frequency range
is used for overall system noise and power measurements 13Sep2020
% Somashekar and Raghavendra October 2020
% Further modification for poly fit & inclusion of Occupancy plotting
============================================================

close all;
clear all;
fb=1; lin=0;EGdB=30; % External amplifier gain in dB
EGLin= 10^(EGdB/10); % external amplifier gain in linear units
avgn=1000; % Number of traces to be averaged for plotting

% PBF1 and PEF1 are any two  bin multiple frequencies for calculating
% the total power in that subband. If the subband numbers are not
correct
% total power of the entire band is calculated and displayed in the
figure
% of Total band power in each trace Vs time in secs/min/hrs
% Choose any one appropiate line or modify to your requirement.
% ------------------------------------------------------------------
----

PBF1 = 120; PEF1 = 400; % For total power in the subband of F1 to F2
within
%PBF1 = 85; PEF1 = 110; % For total power in the FM Band

% ------------------------------------------------------------------
------

% lin=0 for log data

% fb = 0 for full band 20MHz to 1020MHz
% other wise 1 for band defined BF and EF
```

```
% BF is Start Frequenct in MHz , EF is Stop Frequency and SP Step
points
% fname = '2020-07-13_155713';
fname = '2020-10-14-175912a';
cfname = 'G4.csv';

ext1='.csv'; ext2 = '_hdr';
dfname=strcat(fname,ext1);
hfname=strcat(fname,ext2);
% Get as many parameters from the header file....
hf = fopen(hfname);
linedat=fgetl(hf); % Skip first line data
linedat=fgetl(hf); % this has date info.. will include later
linedat=fgetl(hf); % Has Begin Freq. REQUIRED
tmp=strsplit(linedat,': ');
BF=(str2double(tmp(1,2))/1000000);
clear tmp;

linedat=fgetl(hf); % Has End Freq. REQUIRED
tmp=strsplit(linedat,': ');
EF=(str2double(tmp(1,2))/1000000);
clear tmp;

linedat=fgetl(hf); % Has Center Freq. hence skip this
linedat=fgetl(hf); % Has Freq. Span skip this too
linedat=fgetl(hf); % Has RBW Required info. REQUIRED
tmp=strsplit(linedat,': ');
fres=(str2double(tmp(1,2))/1000000);
clear tmp;

linedat=fgetl(hf); % Has VBW hence skip this
linedat=fgetl(hf); % Has Sweep points REQUIRED.
tmp=strsplit(linedat,': ');
sp=str2double(tmp(1,2));
clear tmp;

linedat=fgetl(hf); % Has Sweep time May be REQUIRED.
%%tmp=strsplit(linedat,': ');
%%st=str2double(tmp(1,2));
%%clear tmp;

linedat=fgetl(hf); % Has data type not required
linedat=fgetl(hf); % Has Begin time May be REQUIRED.
tmp=strsplit(linedat,': ');
Starttime=strrep(tmp(1,2),',','-');
clear tmp;
Endtime=Starttime;

%%linedat=fgetl(hf); % Has End time May be REQUIRED.
%%tmp=strsplit(linedat,': ');
%%Endtime=strrep(tmp(1,2),',','-');
%%clear tmp;


%%BF = 30; EF=230; fres=0.1; % taken as 100KHz
%%sp=101;
```

```
%-----------------------------------------------
% To take care of the Day change.............
%-----------------------------------------------
pwd
a1 = load(dfname);

% C2=load(cfname);
% C1=transpose(C2);
% C1GdB= C1(2,:);
% [r,c] = size(C1GdB);
% for i=1:c
%   C1GLin(1,i) = 10^((C1GdB(1,i))/10);
% end
% clear C1;
% clear C2;

cfname1='G4.csv';
cfname2='G3.csv';
%cfname1='C1.csv';
%cfname2='C2.csv';
C1=load(cfname1);
C2=load(cfname2);
C1=transpose(C1);
C2=transpose(C2);

C1GdB= C1(2,:);
C2GdB=C2(2,:);

[r,c] = size(C1GdB);
for i = 1:c
  if (C1GdB(1,i)-C2GdB(1,i)) > 2.5
     C3GdB(1,i)= C1GdB(1,i);
   else
     C3GdB(1,i) = C2GdB(1,i);
  end
 end
 % to be added for mlab code

for i=1:c
C3GLin(1,i) = 10^((C3GdB(1,i))/10);
end
clear C1;
clear C2;


Daynumber=0; DaySecs=86400; DaysCountSec=0;
[r,c] = size(a1);
% tsec=a1(:,2);
TelapSec(1,1)=0;
for i = 1:(r-1)
  if (a1((i+1),2) < a1(i,2))
    Daynumber = Daynumber + 1;
    DaysCountSec = Daynumber*DaySecs;
  end
  TelapSec((i+1),1)=DaysCountSec + a1((i+1),2) - a1(1,2);
end
```

```
%TelapSec=(a1(:,2)-a1(1,2));
% NNNNNNNNNNNNNNN
% Check here for the next statement?
transpose(TelapSec);
TelapHour=TelapSec/3600;



[r,c] = size(a1);
a=a1(:,3:c); % Remove the first  and second element in each row (For
SA)

[r,c] = size(a);
t1 = [1:r]; k=1; n1 = r;sp=c;
%%if lin == 1
%%  sp=c/2;
%%else
%%  sp = c;
%%end

clear a1;


% for calculating xrange
fbin = (EF-BF)/(sp-1);

for j = 1:(sp)
  x1(j) = (BF+(j-1)*fbin);
end
% *********************************************
if (lin==1)
  % If data is in Real and imaginary as in EOR case
  % To calculate the Magnitude (and may be Phase angle not applicable
for SA)
  % For Linear check units and Change formula

 %  Ymag = a/EGLin; % For constant value.
  for k= 1:(r)
      for j = 1:(sp)
        Ymag(k,j) = a(k,j)/C3GLin(1,j);
        Ymag1(k,j) = a(k,j);
        YmagdB(k,j) = 10*log10(Ymag(k,j))+30; % Check 10 or 20 log10
        YmagdB1(k,j) = 10*log10(Ymag1(k,j))+30; % Check 10 or 20 log10
      end
  end

  %tp(k) = 10*log10(sum((YmagdB(k,1:sp)))) + 30;

else

  % *********************************************
  % If data is in log magnitude as in SA and RFI
  %  YmagdB = a-EGdB; % Constant Subraction
  for k= 1:(r)
      for j = 1:(sp)
```

```
        YmagdB(k,j) = a(k,j)-C3GdB(1,j);
        YmagdB1(k,j) = a(k,j);
        Ymag(k,j) = 10^((YmagdB(k,j)-30)/10);
        Ymag1(k,j) = 10^((YmagdB1(k,j)-30)/10);
        end
  end

end
% ending with Y values in dBm

fbin = (EF-BF)/(sp-1);

%-------------------------------------
% Modified for plotting ONLY the required Frequency Range
% in the next few lines
% ---------------------------------------------
% We will use user defined PBF1 and PEF1 for total power calculation
in the
% Finding corresponding sweep points
input('Begin frequency of the Band (>/= 120 and =/< 400) =  ' );
PBF1= ans;

input('End frequency of the Band (>/= 120 and =/< 400) =  ');
PEF1= ans;

sp1 = ceil(((PBF1-BF)/fbin)+1);
sp2 = ceil(((PEF1-BF)/fbin)+1);

%sp1 = ((PBF1-BF)/fbin)+1;
%sp2 = ((PEF1-BF)/fbin)+1;

% check for invalid Choice of band frequencies Then numbers will
% revert to the actual total band power.

if (sp1< 1) or (sp2 > sp)
  sp1 =1; sp2 = sp;
  PBF1= BF; PEF1 = EF;
end
clear x2;

for j = 1:(sp2-sp1+1)
  x2(j) = (PBF1+(j-1)*fbin);
end
% ***********************************************



% plotting all 1 row for checking
figure(1);
% plot(x1,YmagdB(1,:));
plot(x2,YmagdB(1,sp1:sp2));
hold on;
% plot(x1,YmagdB1(1,:));
% plot(x2,YmagdB1(1,sp1:sp2));
ylabel('Power  in dBm');
xlabel('Frequency in MHz');
```

```
title([fname, '  RFI  characterestics at test Location I Trace']);
grid on;
set(gca,'FontSize',18,'fontWeight','bold');

% junk = kbhit ();
pause(5);


% plotting for small average repeats
hold on;
ylabel('Average power steps from start to end in dBm ');
xlabel('Frequency in MHz');
title([fname, '  RFI  characterestics at test Location ']);
grid on;
set(gca,'FontSize',18,'fontWeight','bold');


Q=((r-mod(r,avgn))/avgn);
for i = 1:Q;
  for k= 1: (sp)
    tavg(k) = 10*log10((sum(Ymag(((((i-
1)*avgn)+1):(avgn*i)),k)))*(1/avgn)) + 30;
    tavg1(k) = 10*log10((sum(Ymag1(((((i-
1)*avgn)+1):(avgn*i)),k)))*(1/avgn)) + 30;
%    tmax(k) = max(YmagdB((((i-1)*10+1):(avgn*i)),k));
%    tdiffmax(k)= tmax(k) - tavg(k);
  end
pause(1);
stavg(i,:) = tavg(1,:);
plot(x2,tavg(sp1:sp2));
% plot(x2,tavg1(sp1:sp2));
pause(3);
end

hold off;

% plotting the Average of  all the rows
figure(2);
for k= 1: (sp)
  tavgLin(k) = sum(Ymag((1:r),k)*(1/r));
  tavg(k) = 10*log10((sum(Ymag((1:r),k)))*(1/r)) + 30; % Optimize this
line
  tmax(k) = max(YmagdB((1:r),k));
  tdiffmax(k)= tmax(k) - tavg(k);
  tmin(k) = min(YmagdB((1:r),k));
  tdiffmin(k) = tavg(k) - tmin(k);
end
% plotiing
% plot(x1,tavg);
plot(x2,tavg(sp1:sp2));
%ylabel({'Min, Ave, Max in dBm',' Max-Ave in dB'});
ylabel({'Min, Ave, Max in dBm'});
% ylabel('Min, Ave and (Max-Ave in dB) in dBm');
xlabel('Frequency in MHz');
title([fname, ' RFI  characterestics at test Location ']);
grid on;
```

```
set(gca,'FontSize',18,'fontWeight','bold');
hold on;

%plot(x2,tdiffmax(sp1:sp2));

plot(x2,tmin(sp1:sp2));
plot(x2,tmax(sp1:sp2));


hold off;

% plot with y1 and y2 axis
% plotyy(x1,tmin,x1,tdiffmax);

grid on;
% junk = kbhit ();

pause(2);



% % plotting for total power in the band
% % We will use PBF1 and PEF1 for total power calculation in the
% % Finding corresponding sweep points
%
% sp1 = ceil(((PBF1-BF)/fbin)+1);
% sp2 = ceil(((PEF1-BF)/fbin)+1);
%
% % check for invalid Choice of band frequencies Then numbers will
% % revert to the actual total band power.
% % if sTATEMENT NEEDS TO BE RELOOKed
%
% if (sp1< 1) or (sp2 > sp)
%   sp1 =1; sp2 = sp;
%   PBF1= BF; PEF1 = EF;
% end



for k= 1: r
  for j = (sp1):(sp2)
%  for j = 1:(sp)
    la(k,j) = Ymag(k,j);
  end
  tp(k) = 10*log10((sum((la(k,sp1:sp2))))) + 30;
%  tp(k) = 10*log10((sum((Ymag(k,1:sp))))) + 30;
end
% plot total power for each trace
figure(3);
plot(TelapHour ,tp);
ylabel([{'Total Band Power for each trace in dBm'},[' Band
',num2str(PBF1),' to ',num2str(PEF1),' MHz ']] );
xlabel('Time in Hours');
title([fname, '  Total RFI power for each scan at Location']);
grid on;
set(gca,'FontSize',18,'fontWeight','bold');
```

```
% junk = kbhit ();
pause(2);


%Plotting the rainfall plot
figure(4);
y1=YmagdB(:,1:sp);
%CLIMITS=[-70,-30];
%imagesc(x1,TelapSec,y1,CLIMITS);
imagesc(x1,TelapHour,y1);
colormap('jet');
colorbar;
%imagesc(y1);
ylabel('Time in Hours');
xlabel('Frequency in MHz');
title ([fname, ' RFI measurement at Location and date']);
set(gca,'FontSize',18,'fontWeight','bold');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Occupancy plot FOR COMPLETE SPECTRUM starts fro here.
% Threshold from base line as desired by user
% We have restricted the band of interest from
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



% Continuation code for base line realisation.
% Carry out subaverage based on avgn
%avgn, Q, stavg, Ymagdb ... many variables are from RFI_Pxxx.m
for mm=1:Q
    figure(10);
    clear tnew;
    tnew(1,:)=stavg(mm,:);
    p=polyfit(x2,tnew(sp1:sp2),2)
    plot(x2,tnew(sp1:sp2));
    f1=polyval(p,x2);
    hold on;
    plot(x2,f1);
% hold off;
    csvwrite('FF1.csv',f1);
% csvwrite('YdB.csv',YmagdB(:,sp1:sp2));
    csvwrite('tmin.csv',tnew(sp1:sp2));
    tnew=load('tmin.csv');
    [RR,CC]=size(tnew);
    for i=1:CC
        if(tnew(i)-f1(i)) > 10
            tnew(i)= f1(i);
        end
    end
% First iteration
    figure(10);
    csvwrite('tmin.csv',tnew);
    p=polyfit(x2,tnew,2);
    plot(x2,tnew);
```

```
    f1=polyval(p,x2);
    hold on;
    plot(x2,f1);
    hold off;
    csvwrite('FF1.csv',f1);
    tnew=load('tmin.csv');
    [RR,CC]=size(tnew);
    for i=1:CC
        if(tnew(i)-f1(i)) > 5
            tnew(i)= f1(i);
        end
    end
% pause(3);
% Second iteration
    figure(10);
    csvwrite('tmin.csv',tnew);
    p=polyfit(x2,tnew,2)
    plot(x2,tnew);
    f1=polyval(p,x2);
    hold on;
    plot(x2,f1);
    hold off;
    csvwrite('FF1.csv',f1);
    tnew=load('tmin.csv');
    [RR,CC]=size(tnew);
    for i=1:CC
        if(tnew(i)-f1(i)) > 2
            tnew(i)= f1(i);
        end
    end
% pause(3);
% third iteration
    hold off;
    figure(10);
    csvwrite('tmin.csv',tnew);
    p=polyfit(x2,tnew,2)
    plot(x2,tnew);
    f1=polyval(p,x2);
    hold on;
    plot(x2,f1);
% hold off;
    csvwrite('FF1.csv',f1);
    tnew=load('tmin.csv');
    [RR,CC]=size(tnew);
    for i=1:CC
        if(tnew(i)-f1(i)) > 0.5
            tnew(i)= f1(i);
        end
    end
% Fourth iteration
% pause(3);
    hold off;
    figure(10);
    csvwrite('tmin.csv',tnew);
    p=polyfit(x2,tnew,2)
    plot(x2,tnew);
```

```
    f1=polyval(p,x2);
    hold on;
    plot(x2,f1);
    hold off;
    csvwrite('FF1.csv',f1);
    tnew=load('tmin.csv');
% pause(3);
    f2(mm,:)= f1(1,:);
    clear tnew;
end
% Iteration ends here

OCPcnt=0;
Thold=10;

%%%% for OCPcnt = 1: 10
OCPcnt=1;
while (OCPcnt < 10)
input('RFI Threshold in dB (100 to exit) = ');
Th= ans;% presently not checking the given value


figure(OCPcnt+10);
if Th ==100
    OCPcnt =11;
  end

 for j=1:(sp2-sp1+1)
    OCC(1,j)= 0;
    for i=1:(Q*avgn)
        mm=ceil(i/avgn);
        if YmagdB(i,j+sp1)-f2(mm,j) > Th
            OCC(1,j) = OCC(1,j)+1;
        end
    end
 end
 OCCP=100*OCC/(Q*avgn);
 plot(x2,OCCP);
 ylabel('RFI Occupancy in percentage');
 ylabel([{' RFI Occupancy in percentage '},[' Threshold  >
',num2str(Th), 'dB ']] );
xlabel('Frequency in MHz');
title([fname, '  RFI Occupancy over complete duration ',
num2str(TelapHour(r,1)), ' Hrs ' ]);
grid on;
set(gca,'FontSize',18,'fontWeight','bold');
OCPcnt = OCPcnt+1;
end
% repeated for various threshold ends here

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Occupancy plot FOR SELECTED TIME starts from here.
% Threshold from base line as desired by user
% We have restricted the band of interest from
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Continuation code for base line realisation.
% RFI OCCUPANCY IS TAKEN DURING THE TIME WHEN THE SPECTRAL
% POWER IS REASONABLY CLEAN... OR ONE CAN GIVE
% START AND STOP TIME WHEN ONE WANTS TO FIND OCCUPANCY
% POWER DUE TO HOT AND COLD SKY WILL NOT BE CONSIDERED
% FOR THIS ANALYSIS.
for mm = 1:10

input('Begin Analysis Start time =  ');
STelapHour= ans;

input('End Analysis End time = ');
ETelapHour = ans;

TP1= ceil(((r/TelapHour(r,1))*STelapHour+1));
TP2= ceil(((r/TelapHour(r,1))*ETelapHour));

% Average of  selected rows as per selected time
for k= 1: sp
  Ttavg(k) = 10*log10((sum(Ymag((TP1:TP2),k)))*(1/(TP2-TP1))) + 30;
end


figure(10);
clear tnew;
tnew(1,:)= Ttavg(1,:);
p=polyfit(x2,tnew(sp1:sp2),2);
plot(x2,tnew(sp1:sp2));
f1=polyval(p,x2);
hold on;
plot(x2,f1);
% hold off;
csvwrite('FF1.csv',f1);
% csvwrite('YdB.csv',YmagdB(:,sp1:sp2));
csvwrite('tmin.csv',tnew(sp1:sp2));
tnew=load('tmin.csv');
[RR,CC]=size(tnew);
for i=1:CC
  if(tnew(i)-f1(i)) > 10
     tnew(i)= f1(i);
  end
 end
% First iteration
% pause(5);
figure(10);
csvwrite('tmin.csv',tnew);
p=polyfit(x2,tnew,2);
plot(x2,tnew);
f1=polyval(p,x2);
hold on;
plot(x2,f1);
hold off;
csvwrite('FF1.csv',f1);
tnew=load('tmin.csv');
```

```
[RR,CC]=size(tnew);
for i=1:CC
  if(tnew(i)-f1(i)) > 5
     tnew(i)= f1(i);
  end
end
% pause(5);
% Second iteration
figure(10);
csvwrite('tmin.csv',tnew);
p=polyfit(x2,tnew,2);
plot(x2,tnew);
f1=polyval(p,x2);
hold on;
plot(x2,f1);
hold off;
csvwrite('FF1.csv',f1);
tnew=load('tmin.csv');
[RR,CC]=size(tnew);
for i=1:CC
  if(tnew(i)-f1(i)) > 2
     tnew(i)= f1(i);
  end
end
% pause(5);
% third iteration
hold off;
figure(10);
csvwrite('tmin.csv',tnew);
p=polyfit(x2,tnew,2);
plot(x2,tnew);
f1=polyval(p,x2);
hold on;
plot(x2,f1);
% hold off;
csvwrite('FF1.csv',f1);
tnew=load('tmin.csv');
[RR,CC]=size(tnew);
for i=1:CC
  if(tnew(i)-f1(i)) > 0.5
     tnew(i)= f1(i);
  end
end
% Fourth iteration
% pause(5);
hold off;
figure(10);
csvwrite('tmin.csv',tnew);
p=polyfit(x2,tnew,2);
plot(x2,tnew);
f1=polyval(p,x2);
hold on;
plot(x2,f1);
hold off;
csvwrite('FF1.csv',f1);
tnew=load('tmin.csv');
```

```
% pause(10);
f2(1,:)= f1(1,:);
clear tnew;


OCPcnt=0;
Thold=10;


OCPcnt=1;
while (OCPcnt < 10)
input('RFI Threshold in dB (100 to exit) =  ');
Th= ans;% presently not checking the given value

figure(OCPcnt+20);
  if Th ==100
    OCPcnt =11;
  end

 for j=1:(sp2-sp1+1)
    TOCC(1,j)= 0;
    for i=TP1:TP2
%        mm=ceil(i/avgn);
        if YmagdB(i,j+sp1)-f2(1,j) > Th
            TOCC(1,j) = TOCC(1,j)+1;
        end
    end
 end
 TOCCP=100*TOCC/(TP2-TP1+1);
 plot(x2,TOCCP);
 ylabel([{'RFI Occupancy in percentage'},[' Threshold  >
',num2str(Th), ' dB ']] );
xlabel('Frequency in MHz');
title([fname, ' RFI Occupancy from ', num2str(STelapHour), 'Hrs to ',
num2str(ETelapHour), 'Hrs']);
grid on;
set(gca,'FontSize',18,'fontWeight','bold');
OCPcnt = OCPcnt+1;
end
% repeated for various threshold ends here

end
% repeated time Analysis ends here


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Replotting the power levels and colour plot for different subband
% as desired by user
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

C01=5;
for B1 = 1:10

% plotting for total power in the band
% We will use user defined PBF1 and PEF1 for total power calculation
in the
```

```
% Finding corresponding sweep points
input(' Begin frequency of the Band =  ');
PBF1= ans;

input(' End frequency of the Band =  ');
PEF1= ans;


sp1 = ((PBF1-BF)/fbin)+1;
sp2 = ((PEF1-BF)/fbin)+1;

% check for invalid Choice of band frequencies Then numbers will
% revert to the actual total band power.

if (sp1< 1) | (sp2 > sp)
  sp1 =1; sp2 = sp;
  PBF1= BF; PEF1 = EF;
end
clear x2;

for j = 1:(sp2-sp1)
  x2(j) = (PBF1+(j-1)*fbin);
end
% ********************************************

for k= 1: r
  for j = (sp1):(sp2)
%  for j = 1:(sp)
    la(k,j) = Ymag(k,j);
  end
  tp(k) = 10*log10((sum((la(k,sp1:sp2))))) + 30;
%  tp(k) = 10*log10((sum((Ymag(k,1:sp))))) + 30;
end
% plot total power for each trace
figure(C01);
plot(TelapHour ,tp);
ylabel([{'Total Band Power for each trace in dBm'},[' Band
',num2str(PBF1),' to ',num2str(PEF1),' MHz ']] );
xlabel('Time in Hours');
title([fname, '  Total RFI power for each scan at Location']);
grid on;
set(gca,'FontSize',18,'fontWeight','bold');

% junk = kbhit ();
pause(2);

C01=C01+1;
%Plotting the rainfall plot
figure(C01);
y1=YmagdB(:,sp1:(sp2-1));
% Try to find the climits from data and use it here
%*****************************************
CL0 = (min(y1));
CL1= ceil(min(CL0))-1;
clear CL0;
CL0 = max(y1);
```

```
CL2 = (ceil(max(CL0)));
CLIMITS=[CL1,CL2];

imagesc(x2,TelapHour,y1,CLIMITS);
%imagesc(x2,TelapHour,y1);
colormap('jet');
colorbar;
%imagesc(y1);
ylabel('Time in Hours');
xlabel('Frequency in MHz');
title ([fname, '  RFI measurement at Location and date']);
set(gca,'FontSize',18,'fontWeight','bold');
C01=C01+1
end
% Multiple band plotting ends here
```

==========