Table of Contents

RDS receiver

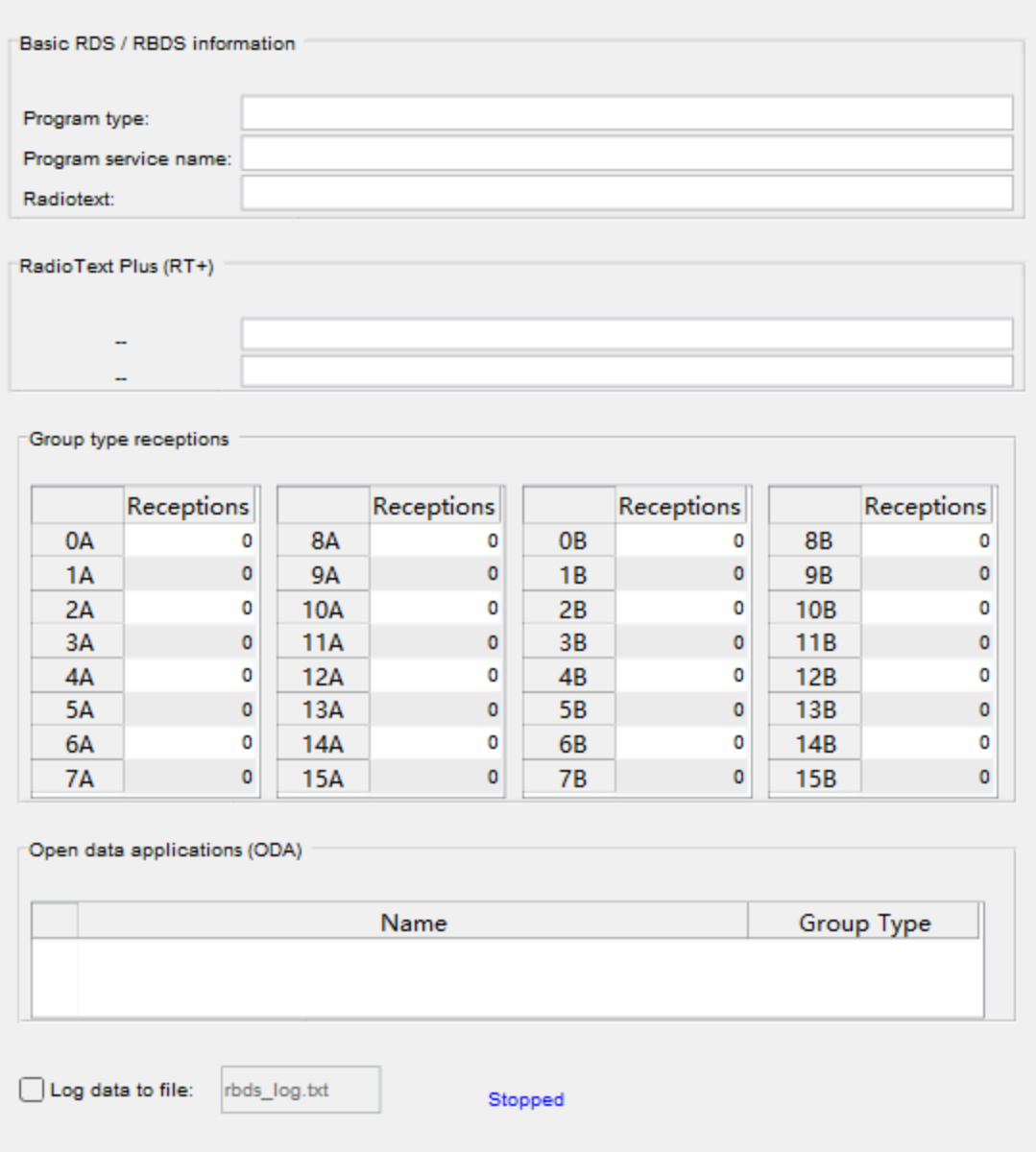
```
%RTL_RDS
%
% No stereo, no de-emphasis filter
% By R.W.

clear all, close all

% Add Mathworkds RDS receiver files in the path, assuming it is a subfolder
% of the current one
if exist('RBDSExample') ~= 2
    addpath ./rds_files
end

% Front-end sampling frequency and decimation factor. Can be changed at will
% This may depend on the type of the dongle:
% SampleRate R must conform to the following conditions: 225e3 < R <= 300e3
% or 900e3 < R <= 3200e3
% RDS symbol rate is D=1187.5 Hz
% RDS is around 57 KHz = 1187.5*16*3
FESR = 1187.5*16*3*4;
% Decimate first by NDEC so that matched fitler operates with L times
 oversampling
% w.r.t. 1187.5 Hz
NDEC = 12;
L = 16;

% Script to define some essential parameters and functions you need to
% figure yourself. The script is included as a p-code file.
% Design a low-pass filter, and the matched filter
% Function to shift frequency
[FRDS,MANMF] = rw_ini_rds(FESR,L);
```

Radio parameters

```
% YLE 1
%expFreq = 87.9e6;
% YLE Puhe
expFreq = 103.7e6;
% YLE Radio Suomi
%expFreq = 94e6;
% HitMix, from Pasila tower
%expFreq = 90.3e6;
```

```
% #Samples read at one round. Matlab example reads only 20 bits at a time
% The number of samples has to be multiple of NDEC and L such that the
% down-sampled vector is always the same size irrespective of the timing.
% Otherwise, Layer 2 and 3 object raise an error.
nSample = NDEC*L*24;
% Number of vectors/frames to read
nFrame = 1e3;

% Your dongle's frequency offset correction here
% Parts per million = frequency_offset/carrier_frequency * million
% Must be integer
% Without a reasonable PPM estimate RDS receiver does not work, because tghe
% bandwidth of RDS is very narrow.
% PPM =62;
PPM = 64;
%PPM = 39;
PPM = 0;

hSDRrRx = comm.SDRRTLReceiver(...
    'RadioAddress', '0',...
    'CenterFrequency',     expFreq, ...
    'EnableTunerAGC',      true, ...
    'SampleRate',          FESR, ...
    'SamplesPerFrame',     nSample, ...
    'FrequencyCorrection', PPM, ...
    'OutputDataType',      'double');
%fprintf('\n')


hSpectrumAnalyzer = dsp.SpectrumAnalyzer(...
    'Name',             'FM signal',...
    'Title',            'FM signal', ...
    'SpectrumType',     'Power density',...
    'FrequencySpan',    'Full', ...
    'SampleRate',       FESR, ...
    'YLimits',          [-50,0],...
    'SpectralAverages', 10, ...
    'FrequencySpan',    'Start and stop frequencies', ...
    'StartFrequency',   -FESR/2, ...
    'StopFrequency',    FESR/2,...
    'Position',         figposition([50 30 30 40]));


hFMDemoSpectrum = dsp.SpectrumAnalyzer(...
    'Name',             'FM demod.',...
    'Title',            'FM demod.', ...
    'SpectrumType',     'Power density',...
    'FrequencySpan',    'Full', ...
    'SampleRate',       FESR, ...
    'YLimits',          [-50,0],...
    'SpectralAverages', 10, ...
    'FrequencySpan',    'Start and stop frequencies', ...
    'StartFrequency',   0, ...
```

```
    'StopFrequency',    FESR/2,...
    'PlotAsTwoSidedSpectrum', false,...
    'Position',         figposition([60 10 30 40]));

% Read and write the input signal
fname = 'rds_in.bb';
hWrite = comm.BasebandFileWriter(...
    'Filename',         fname,...
    'SampleRate',       FESR, ...
    'CenterFrequency',  expFreq,...
    'NumSamplesToWrite',Inf);

hRead = comm.BasebandFileReader(...
    'Filename',         fname, ...
    'SamplesPerFrame',  nSample);

% Constellation diagram
hCon = comm.ConstellationDiagram(...
    'ReferenceConstellation', [-1, 1]);

% Selecct file source or radio source
hSrc = hRead;
%hSrc = hSDRrRx;


% Layer 2 object
rdsDatalinkDecoder = RBDSDataLinkDecoder();

% Layer 3 object. Implementation changed between the two Matlab versions
if strcmp(version('-release'), '2023b')
    rdsSessionDecoder  = RBDSSessionDecoder_2023b();
else
    % This is Matlab 2023a version
    rdsSessionDecoder  = RBDSSessionDecoder();
end

% register processing implementation for RadioText Plus (RT+) ODA:
rtID = '4BD7';
registerODA( ...
    rdsSessionDecoder,rtID,@RadioTextPlusMainGroup,@RadioTextPlus3A);

% Create the data viewer object
viewer = helperRBDSViewer();

% Audio object to listen to the radio
% Max. sampling freuqency depends on the hardware
% Determine a suitable decimation factor NDEC and front-end sampling ratio
% FESR
hAudio = audioDeviceWriter(FESR/NDEC,'BufferSize',ceil(nSample*2/NDEC));
% List available audio outputs
%getAudioDevices(hAudio);
```



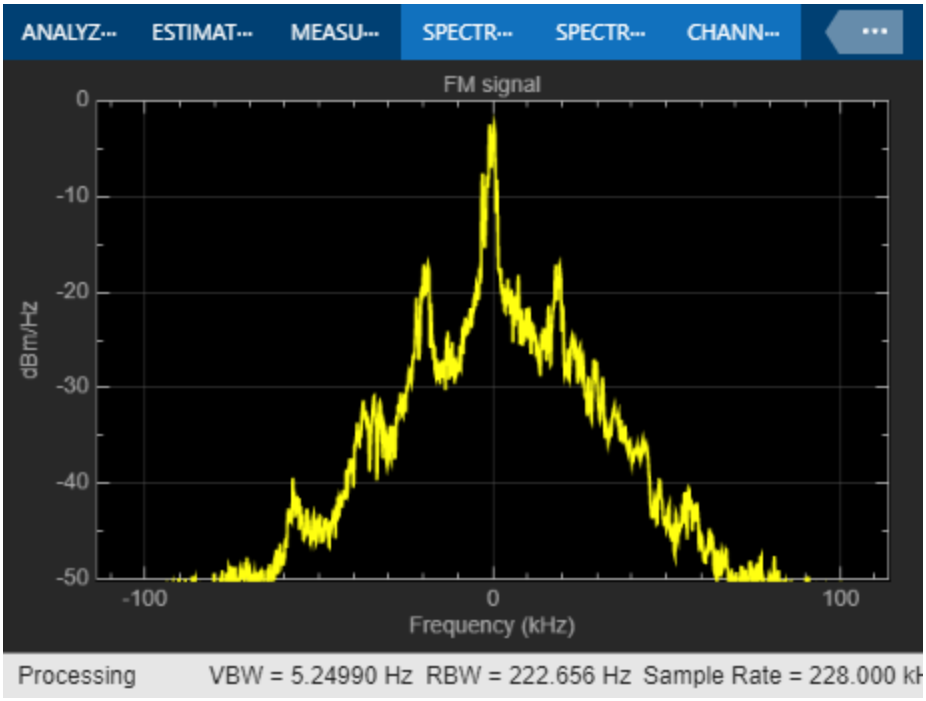
Stream Processing

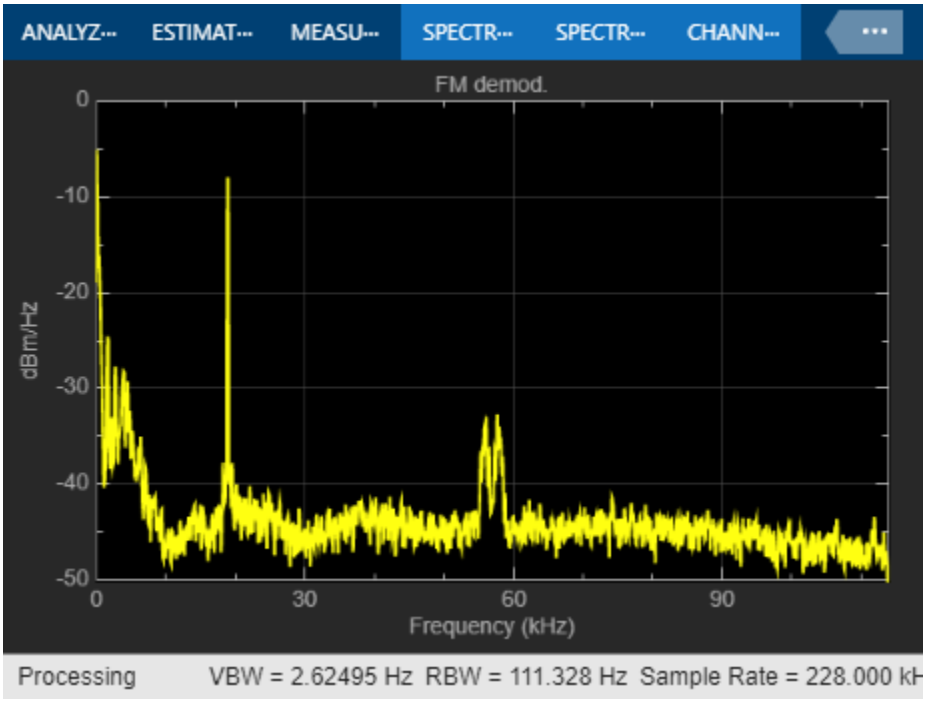
```
%if isempty(sdrinfo(hSDRrRx.RadioAddress))
%    error(message('SDR:sysobjdemos:MainLoop'))
%end

fprintf('Receive time %f [s]   \n', nSample/FESR*nFrame)

% Memory retains the state of the filter between the calls. No notable
% effect
%filter_memory = zeros(1, length(FRDS)-1);

% View the RDS information
start(viewer)
```

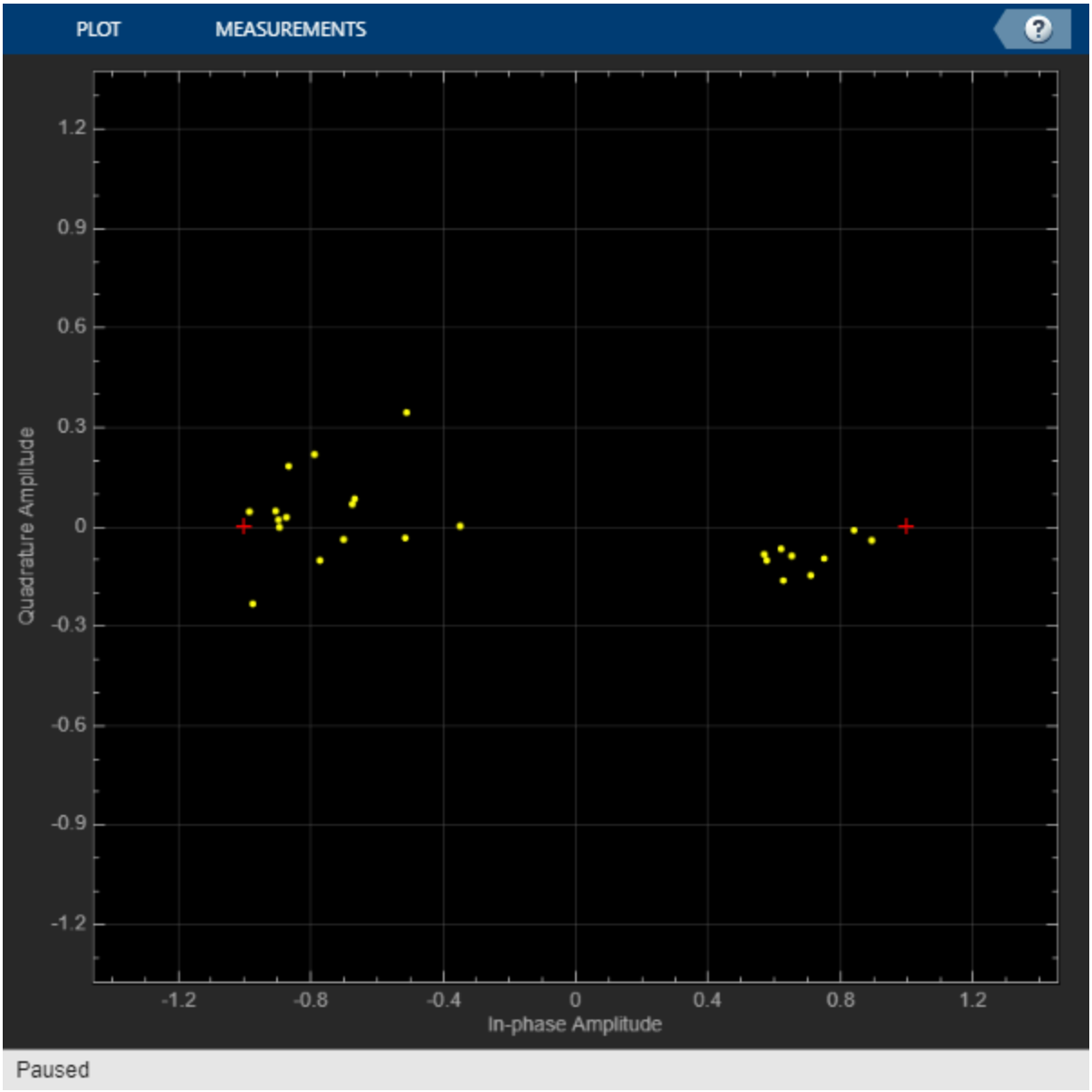
```
% Timing the loop
tic;

% Run as real time as possible. Variables needn't be declared bu don't
% change the size of the array within the loop. Vectors are column ones.

% Store the received signal and write it to the disk after the loop
% This slows down the loop
%rxStore = [];

% Memmory for the differential decoder. Although probably the gap
% between successive calls to read the signal over the air is too large for
% this to have any effect.
diffMem = -1;
% Length of the moving averager in Viterbi&Viterbi
NV=24;
for iFrame = 1:nFrame
    rxSig = hSrc();
    rxSig = rxSig - mean(rxSig);  % Remove DC component
    %rxStore = [rxStore; rxSig];

    % Display received frequency spectrum
    hSpectrumAnalyzer(rxSig);

    % FM demodulation, The output is the same size as input
    fmSig = rw_fmrx(rxSig);

    hFMDemoSpectrum(fmSig);
    % Decimate to listen to the audio
    %lpSig,filter_memory] = filter(FRDS,1,fmSig,filter_memory);
    %aSig = lpSig(1:NDEC:end);

    % Underrun may occure in the loop
    % Arbitrary scaling of the signal amplitude
    %nUnderrun = hAudio(0.5*aSig);
    %if nUnderrun > 0
    %    fprintf('Audio player queue underrun by %d samples.\n',nUnderrun);
    %end

    % Modulate first to baseband
    fm57= rw_shift_freq(fmSig,FESR,-57e3);
    % Filter with a  baseband filter
    fm57 = filter(FRDS,1,fm57);

    % Decimate to L times oversampling w.r.t. to bit rate and L/2 times
    % w.r.t. Manchester code = biphase signal
    rdsDec = fm57(1:NDEC:end);

    % Filter with the filter matched to the biphase pulse.
    % The matched filter (rcosdesign) is root raised cosine, roll-off one, L/2
 samples
    % per symbol minus the same filter delayed by L/4 samples
    rdsMatched = filter(MANMF, 1,rdsDec);
```

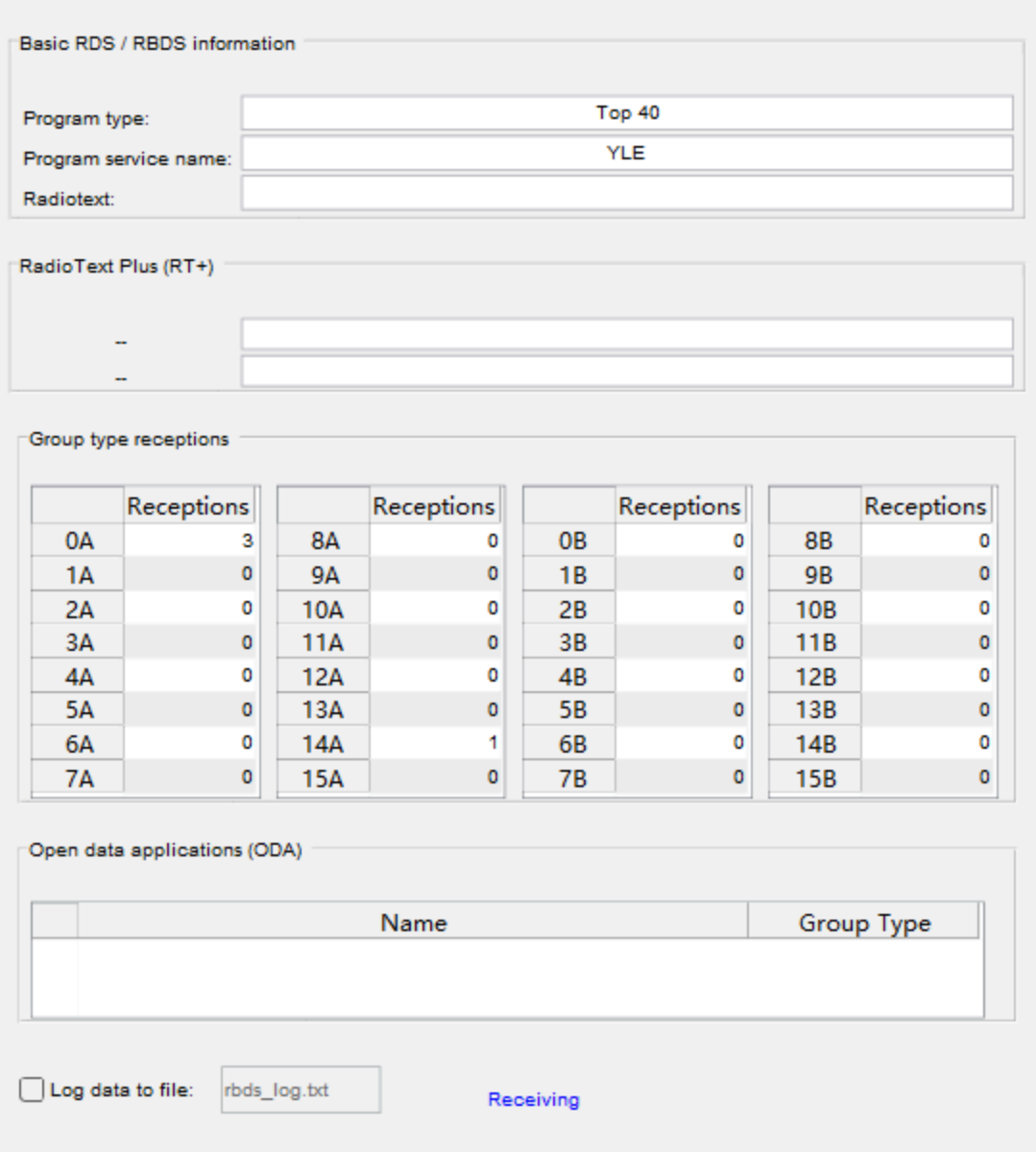
```
    % Search for the timing synch based on max. energy
    % The output is at the symbol rate, 1187.5 Hz, so it is decimated by L
    rdsSym = rw_timing(rdsMatched,L);

    % Viterbi&Viterbi phase offset estimation. The second argument is the
    % length of the moving average. The output is the signal where the
    % phase offset is compensated
    rdsComp = rw_offset(rdsSym,NV);

    % Plot constellation
    hCon(rdsComp);

    % Display constellation. This is faster than the eye diagram
    hCon(rdsComp/max(abs(rdsComp)));

    % BPSK demodulation and differential decoding
    % The decoder operates on complex values and return BPSK {-1,1} symbols
    rdsBit = rw_dbpsk_decode(rdsComp,diffMem) > 0;
    diffMem = 2*rdsBit(end)-1;

    % See the Eye diagram, This considereably slows down the loop
    %if iFrame==1
    %    hnd = eyediagram(rdsMatched,L,1,0,'y-');
    %else
    %    eyediagram(rdsMatched,L,1,0,'y-',hnd);
    %end

    % Process data-link layer (Layer 2)
    [enabled,iw1,iw2,iw3,iw4] = rdsDatalinkDecoder(rdsBit);

    % Process session and presentation layer (Layer 3)
    outStruct = rdsSessionDecoder(enabled,iw1,iw2,iw3,iw4);

    % View results packet contents (Data Viewer)
    update(viewer, outStruct);
end

fprintf('Clock receive time %f [s]\n', toc)
%hWrite(rxStore);
```

```
Receive time 20.210526 [s]
Clock receive time 8.248910 [s]
```


ANALYZ···
ESTIMAT···
MEASU···
SPECTR···
SPECTR···
CHANN···
FM signal
0
-10
-20
-30
-40
-50
dBm/Hz
-100
0
100
Frequency (kHz)
Processing
VBW = 5.24990 Hz RBW = 222.656 Hz Sample Rate = 228.000 kH

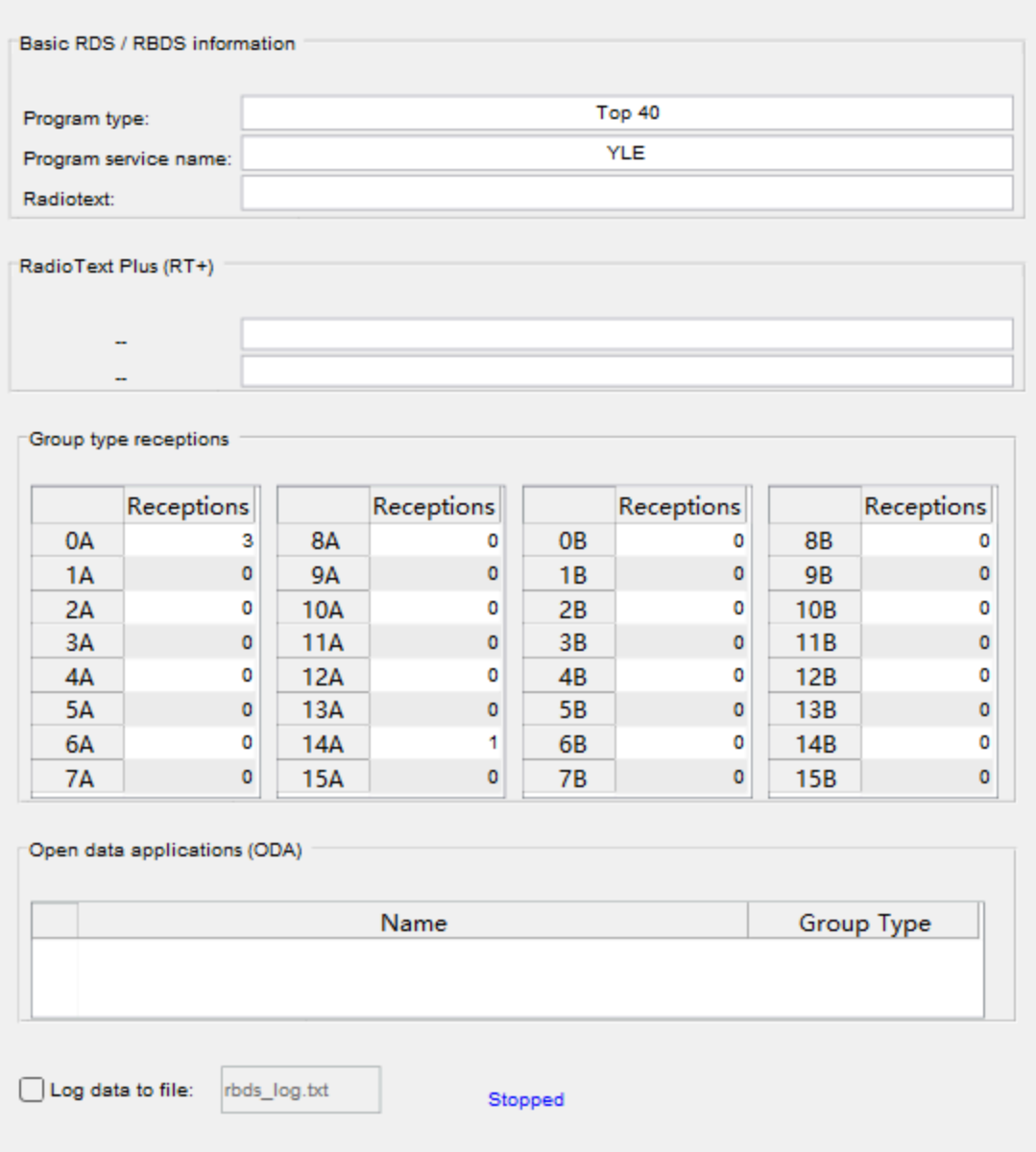

ANALYZ···
ESTIMAT···
MEASU···
SPECTR···
SPECTR···
CHANN···
FM demod.
0
-10
-20
-30
-40
-50
dBm/Hz
0
30
60
90
Frequency (kHz)
Processing
VBW = 2.62495 Hz RBW = 111.328 Hz Sample Rate = 228.000 kH


PLOT
MEASUREMENTS
1.2
0.9
0.6
0.3
0
-0.3
-0.6
-0.9
-1.2
Quadrature Amplitude
-1.2
-0.8
-0.4
0
0.4
0.8
1.2
In-phase Amplitude
Paused



Release all System objects

```
stop(viewer)
release(hSDRrRx);
clear hSDRrRx
release(hAudio);
release(rdsDatalinkDecoder);
release(rdsSessionDecoder);
release(hWrite);
```



Published with MATLAB® R2023a