FT4/FT8 and RX Diversity
for WSJTX 2.1.2

Iztok, S52D

January 24, 2020

Abstract

An implementation of FT8 and FT4 diversity for WSJTX program is described. Time diversity is summing up two consecutive even or odd samples, space diversity is summing up samples from two coherent receivers and two copies of WSJTX, while frame diversity is summing up decoded messages from two WSJTX instances.

Key point for described implementation is how to add up two incoherent audio samples. Solution is to shift one audio sample a few times in time domain, so that every tone gets a possibility to be summed up in phase.

SW installation and operation are described, as well as overview how the code works.

Diversity, as implemented, helps decode more traffic and provides significant improvement in Weak Signal communication on HAM shortwave.

1 Introduction

Diversity reception is present in all modern radio communications. Even cheapest cellular phones have a pair of antennas and receivers. CW and SSB diversity reception is becoming popular (W8JI call it stereo diversity). Simplest form of diversity has two receivers and a logic to select one of them. Properly designed digital protocols like FT4 and FT8 are suitable for better diversity. More advanced techniques are common, where combining is done on bit level and complex processing is used.

FT modes of WSJTX are extremely popular on HAM shortwave bands and we need better reception techniques to deal with interference and other obstacles contributing to weak signal reception. Better FT4 and FT8 decoding is achieved using diversity, by combining two or more RX signal sources. 2 to 5 dB S/N can be gained, enough to make some QSOs possible.

Diversity patch was first implemented in WSJTX 1.9.1 and later in WSJTX 2.0.0 for FT8. FT4 was added in version 2.1.2, as well as optional faster, a bit less efficient, summing of noncoherent samples.
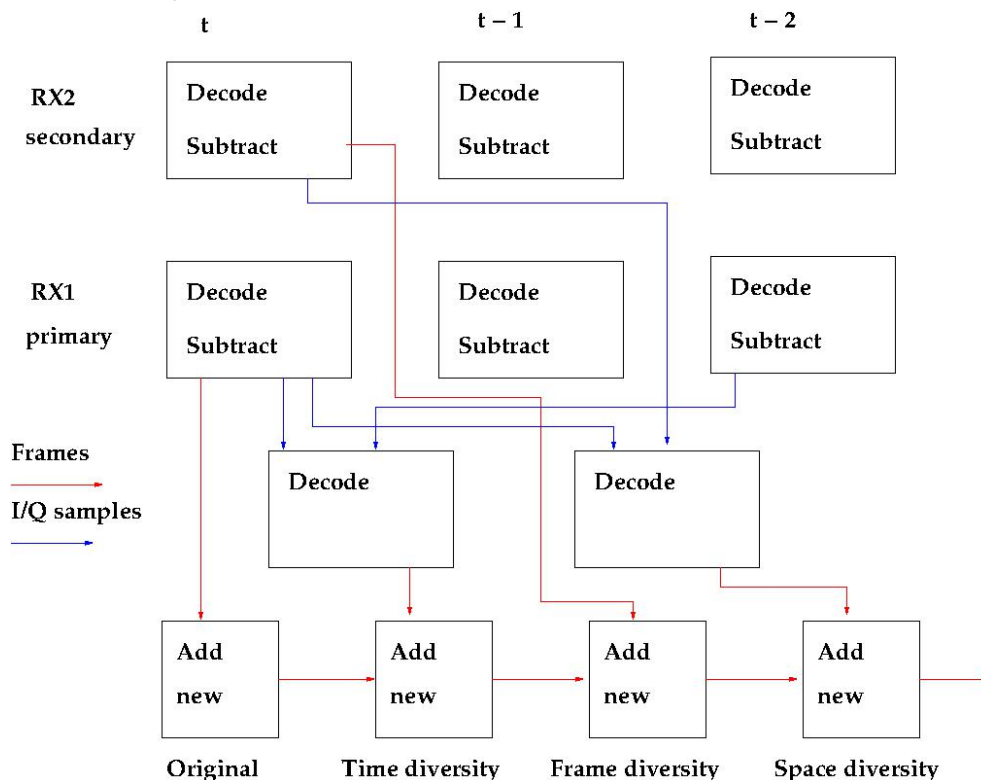
Results are excellent: S52D made several QSOs that would not be possible with standard FT4/FT8 decoder. Benefits are seen on empty band with only few decodes, as well as on crowded 20 m band. Usually 10 % more messages are decoded using both time and space diversity.

2 RX Diversity

Details on different diversity approaches are given, followed by description of how synchronization of noncoherent audio samples was realized.

WSJTX uses multiuser detection: when one frame is properly decoded, it is subtracted from the audio recording, thus new detection is possible in next passes. By combining two residual audio samples (either from different time periods or different receivers) more frames can be decoded.

Though K1JT wrote program MAP65 where two receivers are supporting EME modes, unfortunately it is not usable on HF for FT4 or FT8.

Data flow in WSJTX with diversity. Temporary file is exchanged with decoded frames and residual audio sample.

2.1 Time diversity

Since no additional hardware is needed, any WSJTX user can benefit. Decoder takes two consecutive odd/even samples with residue signals and combines them with proper tone phase. For repeated messages it combines amplitude of the signal, while noise and interference are only power summed.

To put it simply, it can be said that some bits are coming from the current sample, and some from the previous sample.

Sometimes it decodes messages that were transmitted in the previous sample, occasionally both messages are decoded with current time. For example, we can see both RRR and 73 messages together. This happens when summing up with common bits like CALLs.

While operating FT4 or FT8, ″t″ marked messages are not to be trusted completely, as they might be ghosts from previous sample.

Samples are being summed up, while changing bands or just receiver QRG. Thus, old band data can be decoded as being received on the new band and reported to PSKreporter.

2.2 Frame diversity

As implemented, it simply adds messages decoded by secondary WSJTX to the primary one.

Since full decode is done by one decoder, strict RX synchronization is not required and two independent radios can be used.

Both RX have to be tuned to the same frequency, otherwise WSJTX might get confused and transmit on the wrong frequency, as only tone frequency is reported by the secondary copy. Also, when chasing DX, main WSJTX can be tuned to the whole audio band, while secondary is using 200 Hz filers to pick out only the wanted station.

Another possibility is on 160 m: as second WSJTX is listening on JA subband 1908 kHz, there is no need to bother with SPLIT operation, and at the same time it is easy to find clear TX QRG on 1840 kHz.

2.3 Space diversity

For space diversity two copies of WSJTX shall be run, each monitoring same QRG on different receivers attached to different antennas. Both radio receivers must be coherent: it is mandatory to use exactly same filters and same oscillators. Several SDR units (Afedri, Red Pitaya) supports coherent reception, as well as some mainstream RIGs like K3 and IC-7610.

Secondary WSJTX instance writes file to a specified directory, while primary reads it and adds it to decoding. Frame diversity is also used, adding messages decoded by secondary WSJTX and not by primary. File can be properly closed on time as enough delay is provided by time diversity on primary WSJTX.

Real space diversity requires two separated antennas in order to fight fading. Polarization diversity with one vertical and one horizontal antenna gives good results as well.

While testing using IC-7610, benefits were visible by using any pair of antennas. Some testing is needed to get feeling of how space diversity behaves on different bands at different instances.

Though normalization of signal strength was tried, there were no benefits with IC–7610 as both signals are on the same level. Users with two receivers might need to adjust audio levels to get best results.

Files generated by space and frame diversity are named with time, so they must be removed every day to prevent false decoding. Those files written while primary WSJTX was transmitted, or those written too late, are not removed by primary instance.
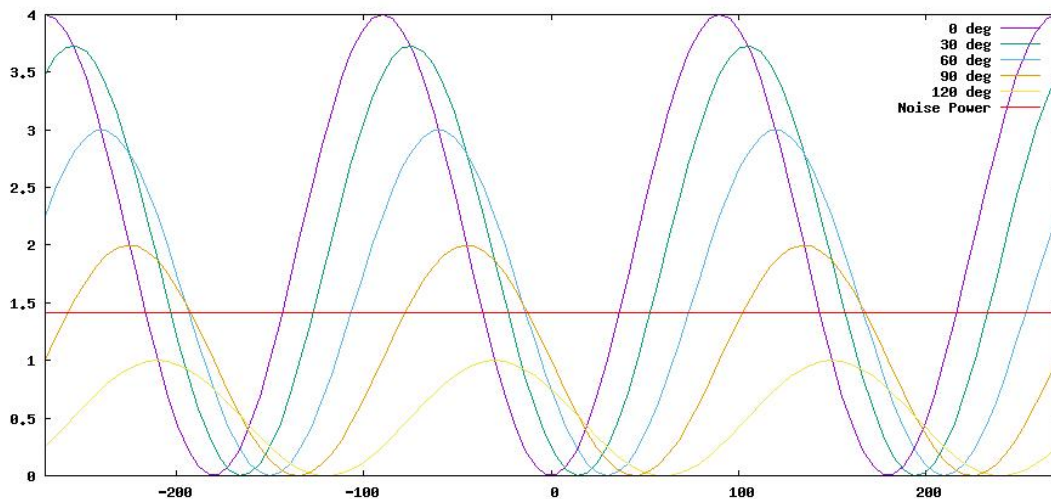
3 Synchronization

Main challenge in diversity implementation was how to synchronize audio samples for different signals. Two instances of WSJTX do not capture audio at exactly same time. TX stations also do not transmit exactly at the same time, the main reason being

the latency of the operating system.

As Audio sample is sampled at 12000 Hz, delay unit is 1/12000 seconds. WSJTX FT4/FT8 decoder subtracts decoded frames from audio sample, and diversity is processing residual audio.

A sequence of 7 delays was found that works fine for audio spectrum from 270 to 3000 Hz. It is based on a simple fact: if two signals are offset 50 degrees, then summing them up when one is delayed 410 degrees, or 770 degrees are same as delaying of 50 degrees. Since FT8 symbol length of 1/6.3 seconds is long enough, small mistakes on symbol transition do not degrade decoding significantly.

Power of two shifted tones: $(sin(\omega t) + sin(\omega t - \phi))^2$ for different phase difference ϕ. Summing up power of two uncorrelated noise sources results in 1.4 times increase in noise power.

Different combinations were tested with tones from 270 to 3000 (2500) Hz with step of 1 Hz and with phase shift 0 to 359 degrees with 1 degree step. When adding two signals with up to 60 degrees shift, degradation from perfect match is marginal, while shift of +/- 90 degrees is still usable.

Tables with best results for 3, 4 and 7 steps are given with percentage of samples where match is found with shift up to given angle.

sequence	bandwidth Hz	30 deg	60 deg	90 deg
0 3 6	270 – 2500	46.4 %	78.1 %	94.1 %
0 3 6	270 – 3000	47.2 %	80.9 %	95.2 %
0 4 8	270 – 2500	48.0 %	81.2 %	95.7 %
0 4 8	270 – 3000	45.4 %	75.5 %	90.4 %
0 7 14	270 – 2500	45.2 %	75.4 %	89.9 %
0 7 14	270 – 3000	45.4 %	77.1 %	91.7 %

Three shifts statistics. 0, 4, 8 is selected.

sequence	bandwidth Hz	30 deg	60 deg	90 deg
0 3 6 10	270 – 2500	57.8 %	88.5 %	98.0 %
0 3 6 10	270 – 3000	57.9 %	89.8 %	98.4 %
0 3 6 9	270 – 2500	58.1 %	87.3 %	97.4 %
0 3 6 9	270 – 3000	58.5 %	89.5 %	97.9 %
0 4 7 10	270 – 2500	57.8 %	88.5 %	98.0 %
0 4 7 10	270 – 3000	57.9 %	89.8 %	98.4 %
0 4 8 12	270 – 2500	60.1 %	91.1 %	98.8 %
0 4 8 12	270 – 3000	56.7 %	85.1 %	94.5 %
0 4 8 17	270 – 2500	58.8 %	90.6 %	99.7 %
0 4 8 17	270 – 3000	55.3 %	84.4 %	94.9 %

Four shifts statistics. 0, 4, 7, 10 is selected.

sequence	bandwidth Hz	16 deg	30 deg	60 deg	90 deg
0 10 17 20 25 30 39	270 – 3000	51.2 %	77.7 %	97.2 %	100.0 %
0 10 17 20 25 31 39	270 – 3000	50.8 %	76.9 %	97.2 %	100.0 %
0 10 20 26 31 39 17	270 – 3000	50.5 %	76.1 %	95.6 %	99.9 %
0 8 15 22 28 31 41	270 – 3000	50.5 %	76.7 %	97.7 %	100.0 %
0 8 15 22 29 31 41	270 – 3000	50.4 %	76.5 %	97.8 %	100.0 %

Seven shifts statistics. 0, 10, 17, 20, 25, 30, 39 is selected.

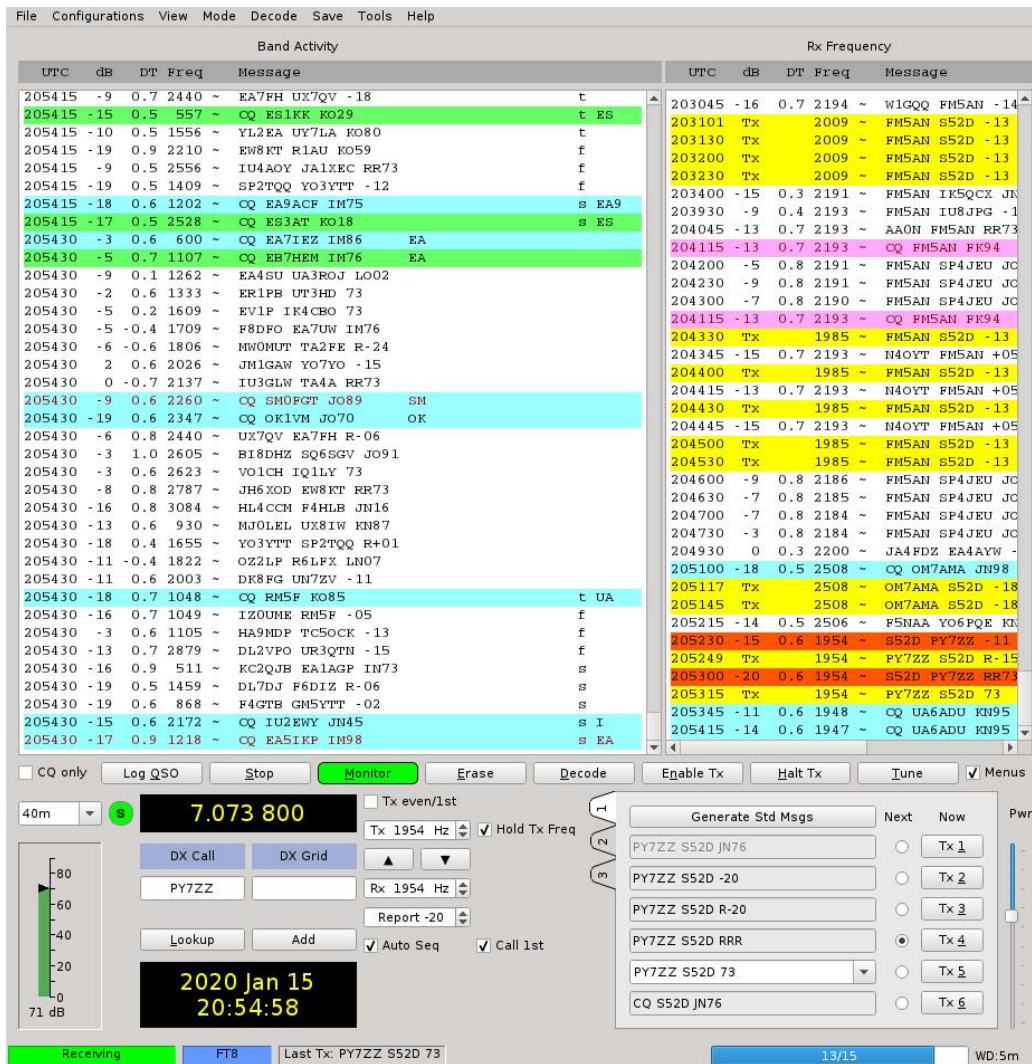# 4   Operation

Just operate as with standard WSJTX.

The only visible change are frames marked with "t/f/s" on the right side, similar as "a" for apriori decoding. They are not shown in Fox/Hound or contest mode, but diversity works. Compound CALLs are sometimes decoded differently in separate steps and both decodes are shown.

Two configuration files are needed to enable data sharing between two WSJTX instances, as well as directory for temporary files. Old files should be deleted once per day manually.

Start with sequence length 7. If CPU can not process frames on time, experiment with faster version of diversity patch using 3 or 4 steps for synchronization. Slightly lower number of messages can be decoded, compared to version with 7 steps.

Use normal decode. If deep decode is selected, decoding might overlap with TX period as the processing time is noticeable longer. Apriori decoding works fine and shall be selected.

Operator must change mode (FT4/FT8) on both WSJTX as mode change is not propagated. VFO–A/VFO–B shall be checked on any QRG change as well.

4.1 Installation

Replace three files (decoder.f90, ft8_decode.f90 and ft4_decode.f90) in src/wsjtx/lib directory and recompile WSJTX. Only jt9 is changed, there are no changes to C++ code. Most users will try time diversity only and it works with no additional configuration files.

Those with two radios and antennas can benefit from frame diversity, while only few of us have possibility to use space diversity. Both demand two copies of WSJTX to run (use -r name), and each needs separate configuration file. Please note: both copies have to be run with option r.

To test setup, start first WSJTX without wsdiv.txt. Time diversity should work. Next, start secondary WSJTX with wsdiv.txt file and check if temporary files are written. If so, it is time to start primary one.

4.2 Configuration file

If there is no configuration file, only time diversity is used.

Files wsdiv.txt shall be created in a writable directory where files ALL.TXT, wsjtx.log and others are stored for each WSJTX instance. On Windows, directories are like

C:\Users\S52D\AppData\Local\WSJT-X -RX2.

File wsdiv.txt has a simple format with 10 numbers and two strings, each on separate line. FT4 and FT8 data are separated, manly because FT4 has shorter gap between RX and TX. For space and time diversity, we need to select length of sync sequence. Valid numbers are 0 (no diversity), 1 (marginal), 3, 4 and 7.

The first 10 lines contain numbers for FT4/FT8 to enable and specify sequence length for time diversity, data writing, data reading for frame diversity and data reading for space diversity. 0 is for disabled, valid values are 1, 3, 4 and 7 for space/time sequence length. Selection is based on CPU power of computer, find a match producing decent results. Decoded frames shall be shown on time, not during TX period.

Since primary WSJTX needs time diversity, the file is read after it was written by secondary WSJTX, thus time diversity and frame diversity are allways working if space diversity is activated.

Common directory where files are stored must be created when two receivers are used. Two directory names are specified, one for writing the file and one for reading. Windows users shall put proper file name, like C:\temp\div\.

Test parameter is not used in this version of the software.

Example for powerful CPU:

meaning	primary	secondary
time diversity FT4	7	3
time diversity FT8	7	0
data writing FT4	0	1
data writing FT8	0	1
frame diversity FT4	1	0
frame diversity FT8	1	0
space diversity FT4	7	0
space diversity FT8	7	0
test parameter FT4	0	0
test parameter FT8 8	0	0
write directory	.	/tmp/div/
read directory	/tmp/div/	.

Example for less powerful CPU. A bit more work is allowed for FT8.

meaning	primary	secondary
time diversity FT4	3	0
time diversity FT8	4	0
data writing FT4	0	1
data writing FT8	0	1
frame diversity FT4	1	0
frame diversity FT8	1	0
space diversity FT4	4	0
space diversity FT8	7	0
test parameter FT4	0	0
test parameter FT8	0	0
write directory	.	/tmp/div/
read directory	/tmp/div/	.

5 Code implementation

Only three files are changed: ft8_decode.f90 and ft4_decode.f90 where real work is done and decoder.f90 where "t/f/s" annotations are generated.

There are no changes to the program structure or to procedure parameters so porting to new versions remains simple.

Brief overlook of code for FT4 (FT8 is similar) is given.

Several new variables are introduced, like:

```
    real ddd(NMAX,5) ! save 4 periods + data read from disk
    integer*2  dttablet(7), dttables(7) ! up to 7 shifts
```

When decoder is run for the first time, wsdiv.txt configuration is read and variables set properly.

```
if(first) then
   open(100,file=trim(data_dir)//'/wsdiv.txt',status='old',err=299)
   first=.false.
endif
```

Data file is opened before normal pass decodes are written.

```
if (fswriteok) then
  open(101,file=trim(divwdir)//datetime//'.dd4',access='stream',status='REPLACE'
  if (ioerr .gt. 0) divsav=.false. ! directory does not exist.
endif
```

There are new passes added to the main loop, and subtraction is activated for all three passes.

```
if (divsav .or. tdivok .or. sdivok) dosubtract=.true. ! if diversity enabled...
do isp = 1,28   ! used to be 1,nsp .10..17 time diversity, 20 frame div, 21-27 s

! isp=10 write dd to file, prepare data for time diversity
 if (isp .eq. 10) then
   write(101) 0,0,0,0,message,iaptype,qual ! mark end of decoded frames
   write(101) dd
   close(101) ! close fast, so other task can read after time diversity
 endif

 if (isp .gt. 10 .and. isp .le. (10+tdlen) ) then
  dtoffset = dttablet(isp-10)
  dd(1:NMAX-dtoffset) = ddd(1:NMAX-dtoffset,ipnow) +ddd(1+dtoffset:NMAX,ipold)
 endif

! here comes space diversity: isp 20 reads files and prepare for space diversity
 if (isp.eq.20) then
   open(102,file=trim(divrdir)//datetime//'.dd4',access='stream',status='old',io
   do
```

```
      read(102,IOSTAT=ioerr,err=199) smax,nsnr,xdt,f1,message,iaptype,qual
      if ((smax .eq. 0.0) .and. (nsnr .eq. 0.0)) goto 197 ! end of frames

      ldupe= ! find if already decoded
      if(.not.ldupe ) then !pass to WSJTX as new decode
         iaptype=iaptype+300  ! frame diversity
         call this%callback(smax,nsnr,xdt,f1,message,iaptype,qual)
      endif
   enddo

! read residual audio from another jt9 instance
197 if (sdivok) then
      read(102,IOSTAT=ioerr,err=199) ddd(:,5)
    endif
 endif

! passes 21 to 27 are for space diversity
 if (isp .ge. 21 .and. isp .le. (20+sdlen) ) then
   dtoffset = dttables(isp-20)
   dd(1:NMAX-dtoffset) = ddd(1:NMAX-dtoffset,ipnow) +ddd(1+dtoffset:NMAX,5)
 endif !isp 20..27
```

For normal passes, decodes shall be saved to file:

```
   if (divsav .and. (isp.lt.10)) then ! frame diversity, s52d
      write(101) smax,nsnr,xdt,f1,message,iaptype,qual
   endif
```

New decodes after diversity are marked as t/s:

```
  if (isp .lt. 20) then
     iaptype=iaptype+100 ! time diversity
  else
     iaptype=iaptype+200 ! space diversity
  endif
```

6   Future work

Described diversity is simplest possible from programming point of view, but demanding for CPU power.

As WSJTX decoder is working with power spectra, we might get as good results by summing up power in the frequency domain (cx variable in sync8 for FT8). This might give similar results with noticeable less computation.

More integration into WSJTX would move configuration file into standard WSJTX configuration menu. By propagating exact RX QRG to decoder we can stop time diversity during QRG change. WSJTX is well balanced and works fine on a different range of computers. Diversity patch is CPU hungry and works only on top range CPUs.

Space diversity can be made simpler, if WSJTX is capturing audio samples at exactly the same time, Modifying WSJTX to work with two radios at the same time is not likely due to complexity of work to cover all possible combinations.

Unlike JT65 averaging, this implementation works on audio sample level. Other modes might benefit as well. If diversity is incorporated into mainstream package, then some additional parameters can be added to decoder functions, enabling a more universal, readable and flexible approach.

External SDR programs can be modified to feed multiple standard WSJTX instances with proper data. This is the way to handle 4 or more receivers configured for space diversity.

7 Conclusion

WSJTX decoders are on the limit of what is possible with single receiver. the effects of using diversity on daily activity is positive, the patch deserves to enter mainstream code in some later release.

HAMs with single receiver can benefit from time diversity, while those with two might benefit from frame diversity.

Beside several QSOs including rare DX by using diversity, author learned a lot by studying WSJTX code and enjoyed every step of it.

8 Acknowledgments

WSJTX development team K1JT, W9AN and G4WJS not only put HAM radio into 21. century, but also opened up their SW, so that work like diversity can be done.

W9MDB compiled Windows version, and beta testing was done by S51TC and K7MDL and KB8O.

Nandu tried to translate Angleščina to English.