The “CODIF” DiFX branch

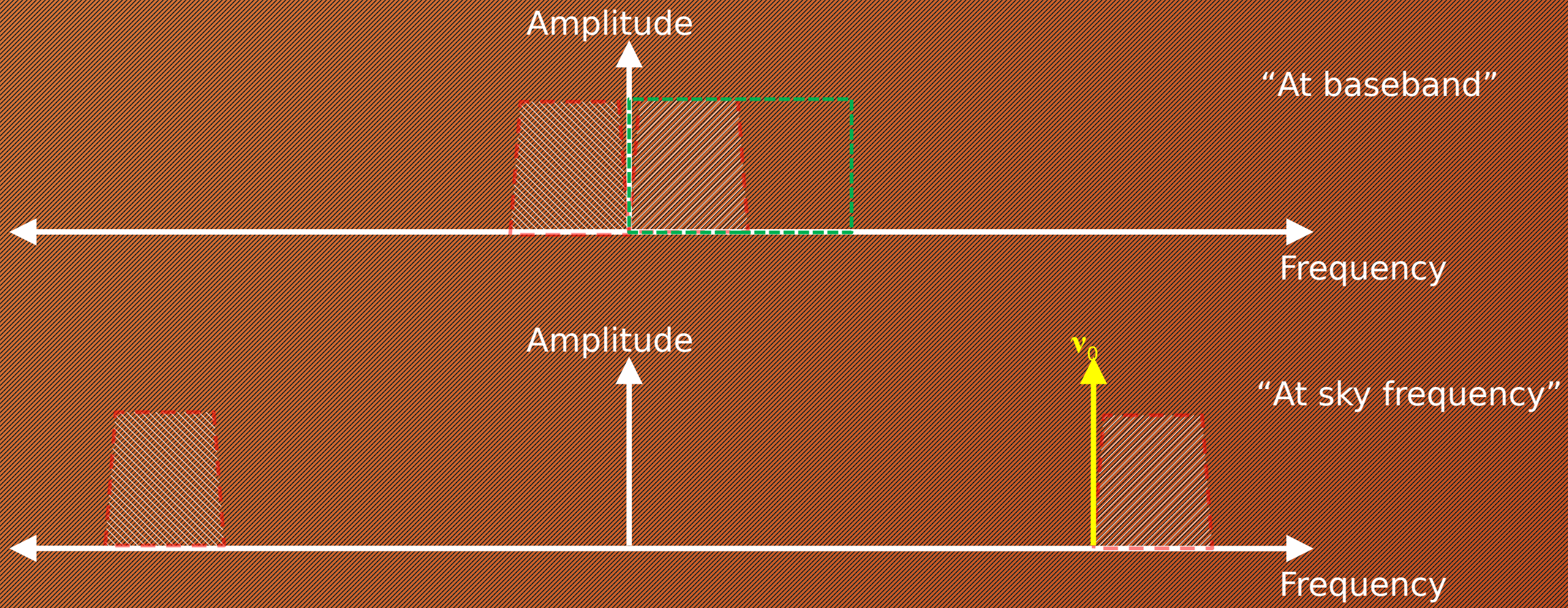
Extensions and changes to support complex oversampled subbands

Adam Deller, 12th DiFX Workshop, Bad Kötzting, Germany 3 Sep 2018

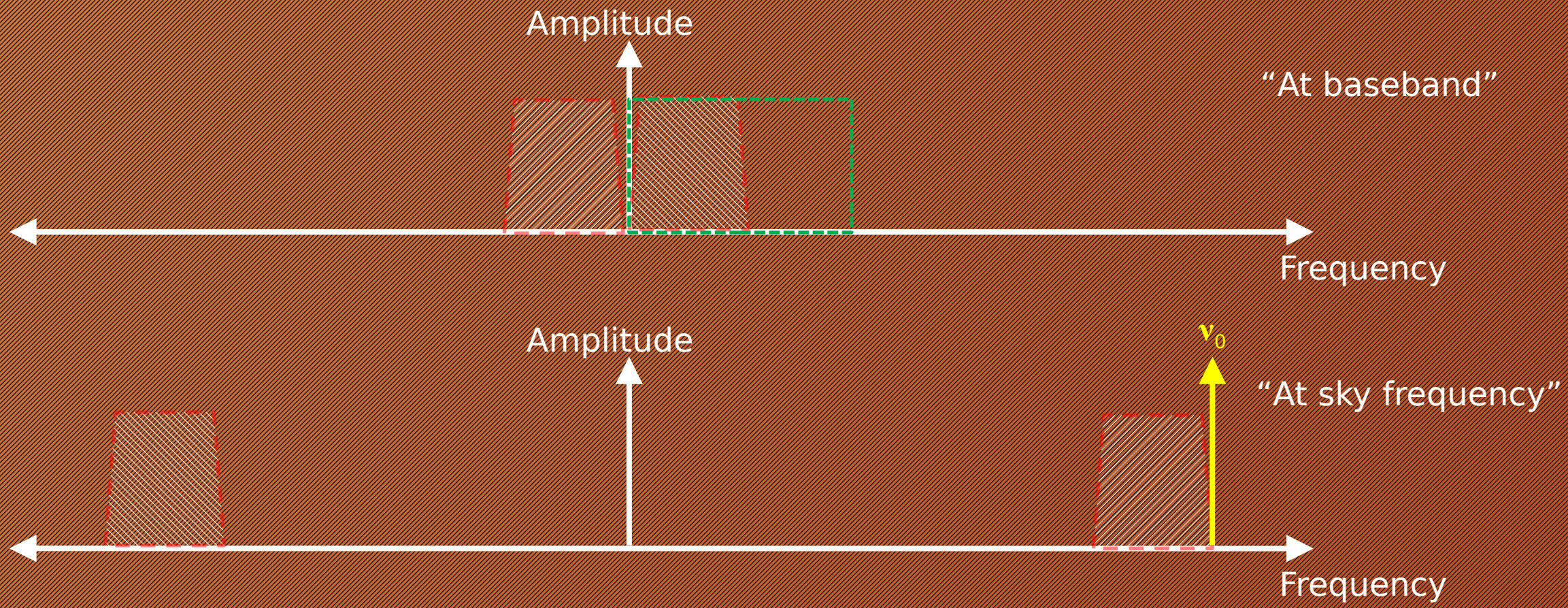
What is a complex band?

- ”Complex” subband has non-zero imaginary component to its voltage representation
- Real subbands have negative frequency components that are the complex conjugate of their positive frequency components (so imaginary part of voltage is always zero).

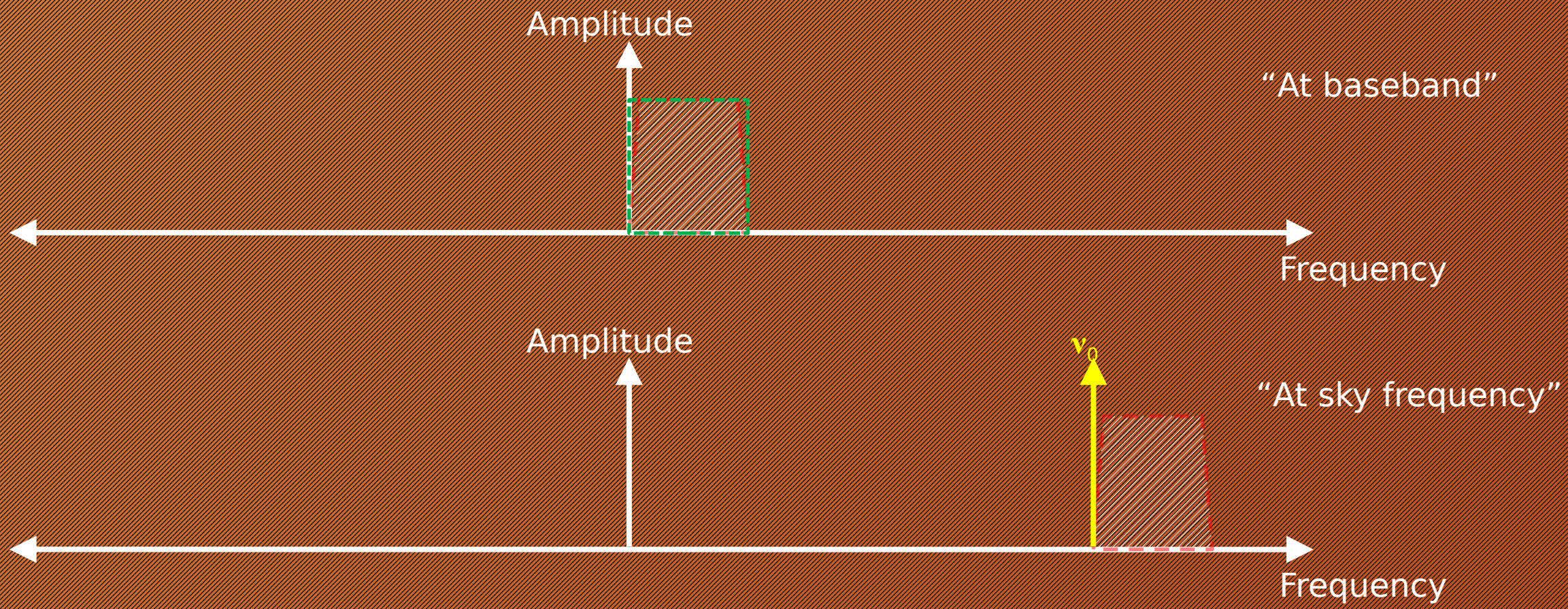
Real-valued upper sideband



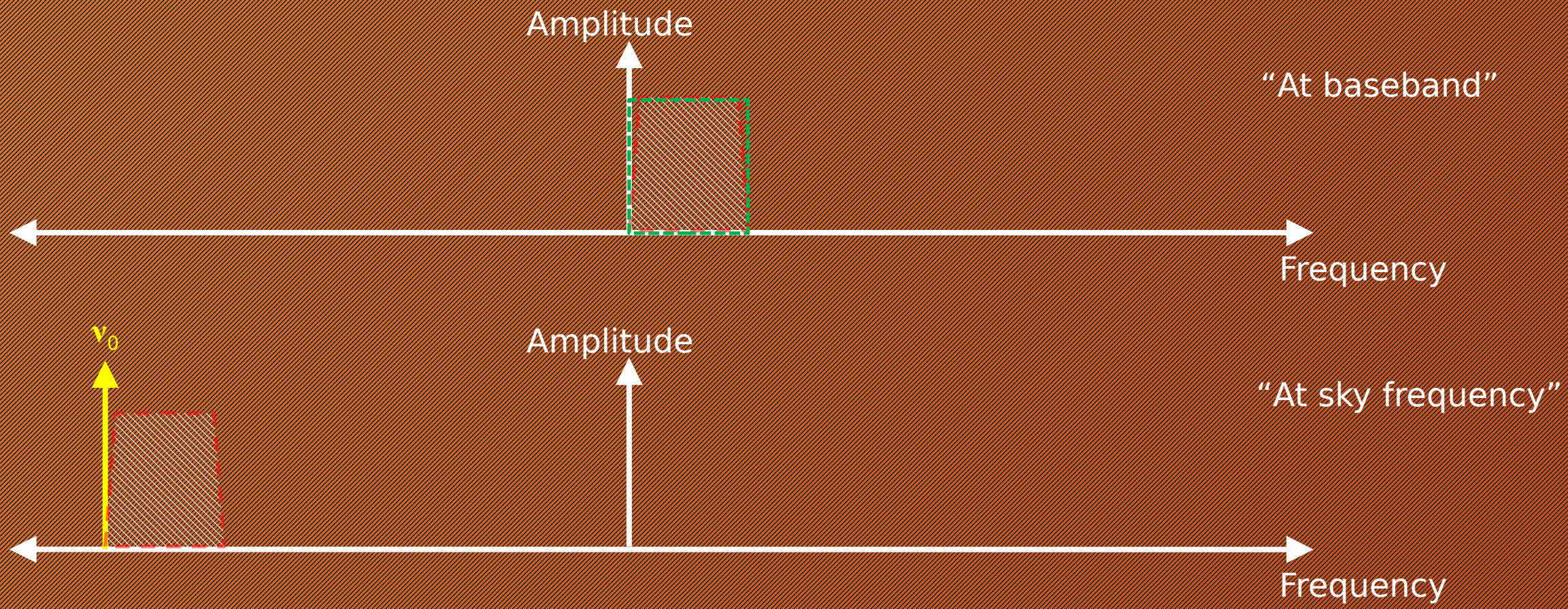
Real-valued lower sideband



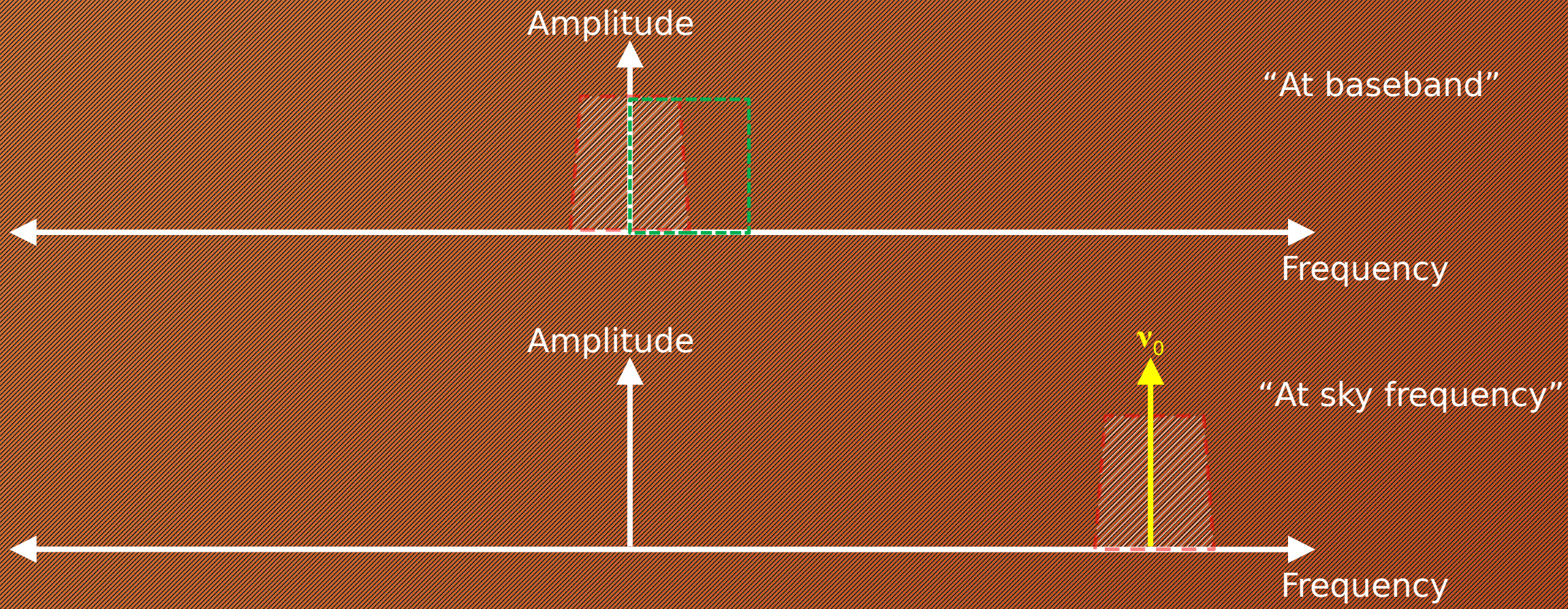
Complex-valued single upper sideband



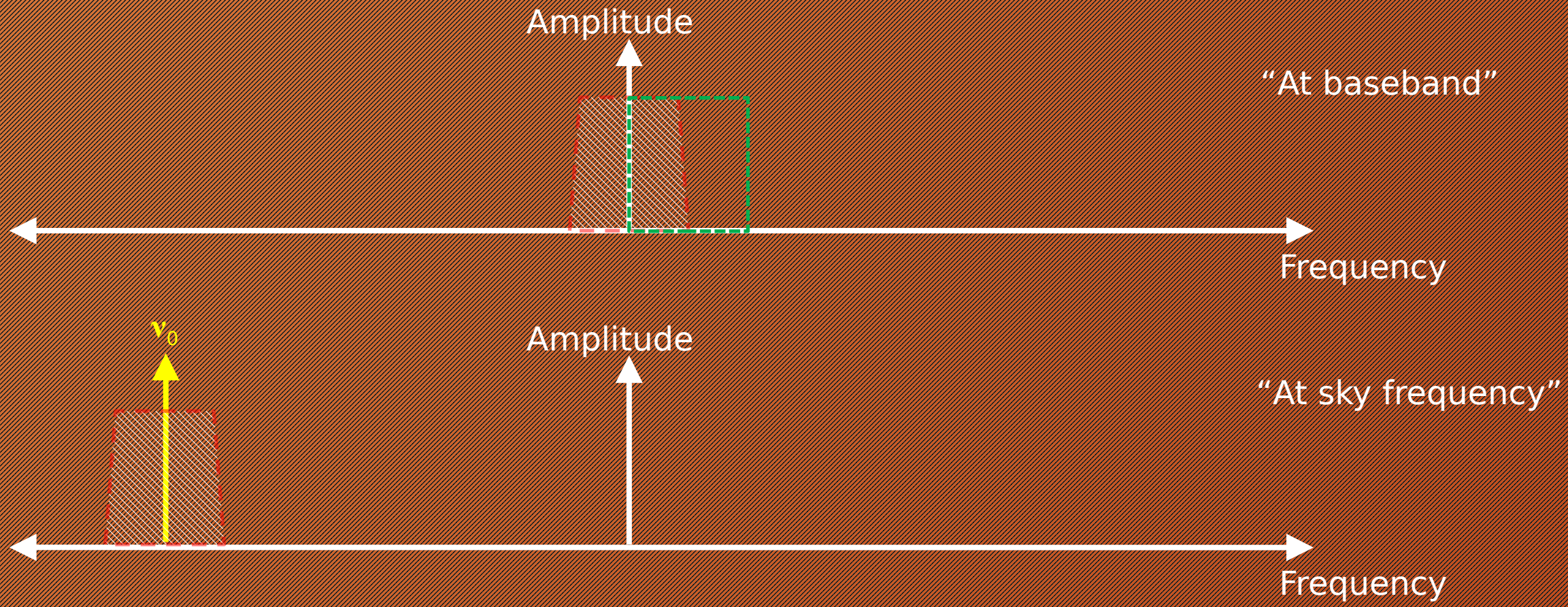
Complex-valued single lower sideband



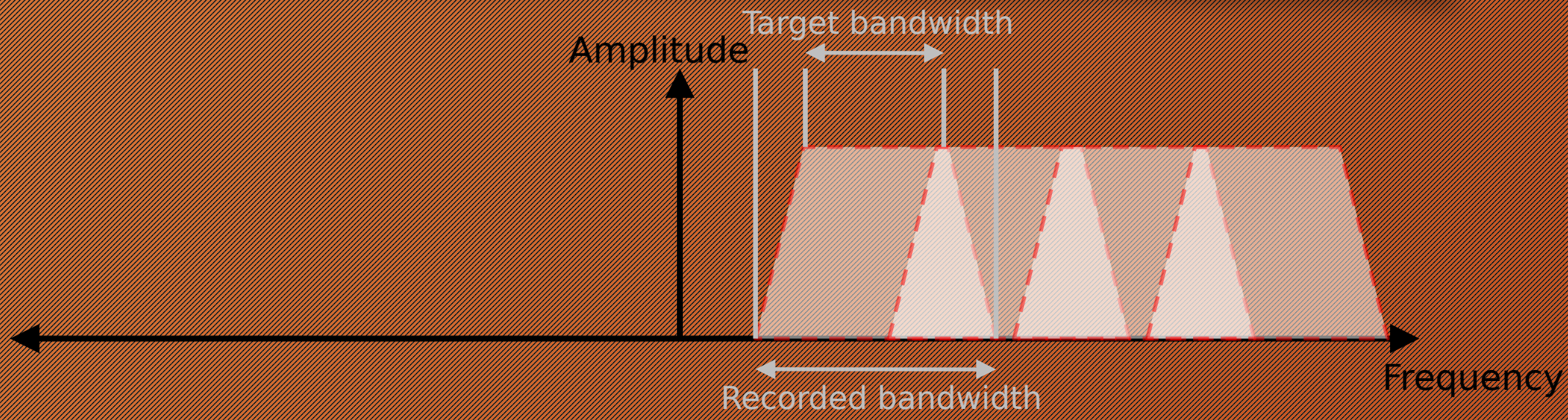
Complex-valued “double upper” sideband



Complex-valued “double lower” sideband



What is an oversampled band?



- **BUT…** the point of view of DiFX, subbands are all independent so DiFX doesn’t care about “oversampling” (DiFX just needs to know the sampled bandwidth)

Impacts on DiFX

- Typical oversample rates (4/3, 8/7, 10/9 etc) demand non-power-of-2 Fourier Transform block sizes in order to get sensible duration blocks (in integer nanoseconds)
- Overlapping bands might confuse downstream processing
  - But a new utility has been added to allow oversampled bands to be stitched together (sites/ASKAP/mergeOversampledDiFX.py, based on Jan's difx2difx)
  - This operates on .difx and .input files to "re-shuffle" visibility channels
- Need to deal correctly with complex-sampled data in 3 forms (lower, upper, double)

How to support complex bands

- Data has to come in in a recognized format
  - VDIF and CODIF support complex-sampled data
  - “Somewhat tested” versions include 1+1 bit CODIF, 4+4 bit CODIF, 8+8 bit CODIF, and (I think) 2+2 bit VDIF
  - Complex lower double sideband most tested, upper single sideband should be safe
- Vex2difx needs to know that the samples are stored as complex
  - Inside ANTENNA block: “`sampling = COMPLEX`” (or “`sampling=COMPLEX_DSB`”)

How to support oversampled bands

- Currently only available in the “CODIF” branch (needs to be merged back in to trunk)
- Current vex standard can’t really support non-unity-denominator oversampling so post-processing of the vex file and hard-coded assumptions are presently an unfortunate necessity
- Tell vex2difx explicitly all about the oversampling: format name like (for complex data, oversampled with 27 second period, 8064 bytes payload + header, 1+1 bit data): `format=CODIFC/27/8064/1`

Case study: ASKAP

- ASKAP produces 300 channels:
  - Each 32/27 MHz (1.185185….) MHz wide (recorded bandwidth), initially in “vcraft” format (binary dump off the beamformer voltage buffer)
  - Spaced at 1 MHz
  - Complex double sideband
  - Can be ”upper” or ”lower” (positive frequency sense or negative frequency sense) depending on observing frequency chosen

Case study: ASKAP

- Workflow:
  - Obtain the vcraft voltage files and “fcm” file describing antennas positions, etc. Then:
  - **`vcraft2obs.py *.vcraft`**
    (produces codif files, obs.txt, chandefs.txt, run.sh)
  - **`askap2difx.py fcm.txt obs.txt chandefs.txt --ants=ak08,ak18,ak25…`**
    (produces a fake .vex file, modifies it, and then uses vex2difx to create usual .input, .calc, .im files, plus .stitchconfig)
  - **`./run.sh (runs the correlation)`**
  - **`rm -rf craftfrb_1D2D.* ; mergeOversampledDiFX.py craftfrb_1.stitchconfig craftfrb_1.difx`**
  - **`difx2fits craftfrb_1D2D`**

Case study: ASKAP

```
adeller@ar313-adeller trunk SB1640> cat obs.txt
startmjd     = 58254.581946371
stopmjd      = 58254.582062112
srcname      = CRAFTSRC
srcra        = 12:30:49.41869616
srcdec       = +12:23:27.9421652
```

```
adeller@ar313-adeller trunk SB1640> cat chandefs.txt
1174 L 1.185185185185185185
1175 L 1.185185185185185185
1176 L 1.185185185185185185
1177 L 1.185185185185185185
1170 L 1.185185185185185185
1171 L 1.185185185185185185
1172 L 1.185185185185185185
1173 L 1.185185185185185185
```

From .v2d file:

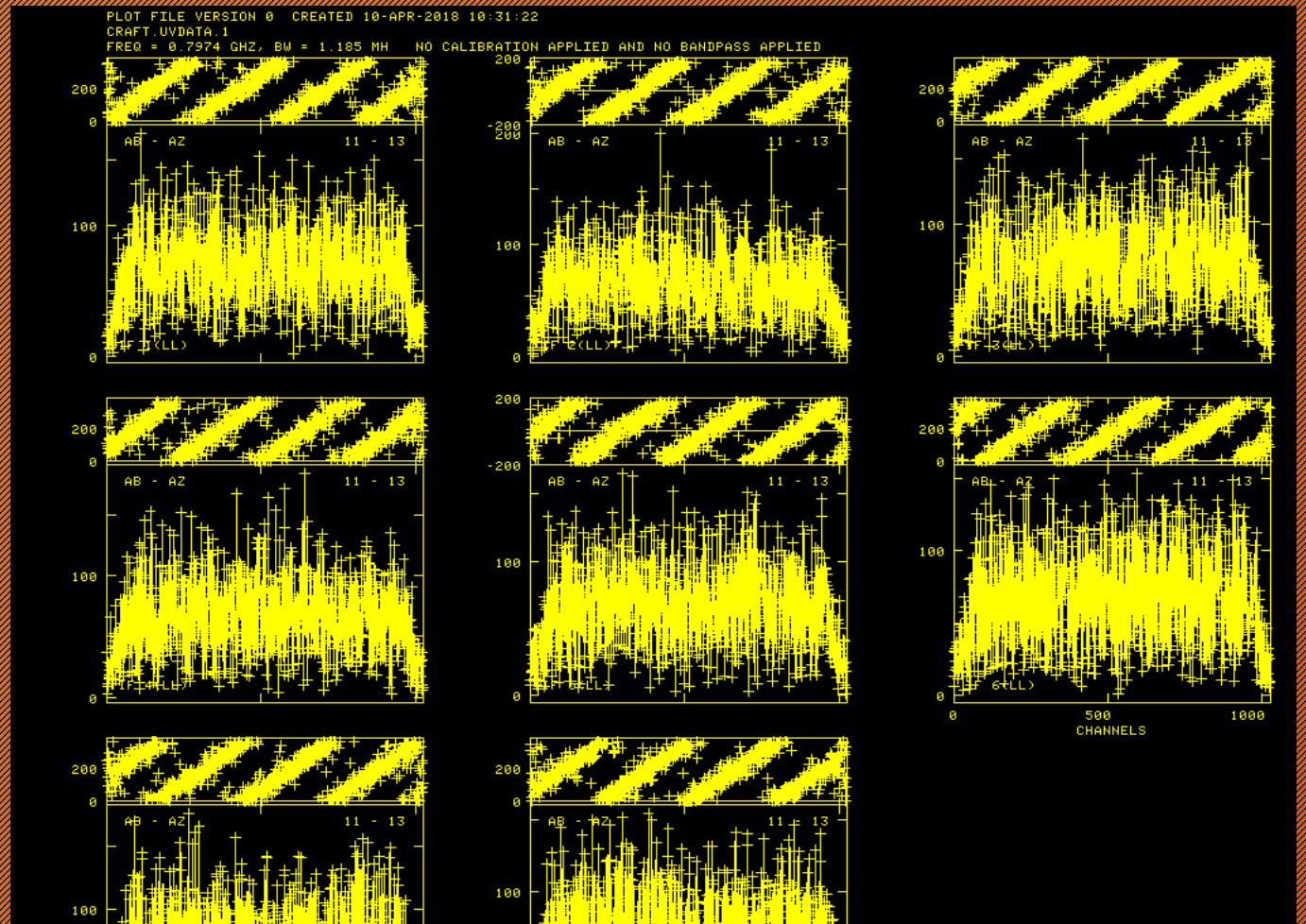
```
ANTENNA A2
{
  file = /home/adeller/askap/ak08.codif
  format=CODIFC/27/8064/1
  phaseCalInt=0
  toneSelection=none
  sampling=COMPLEX_DSB
}
```

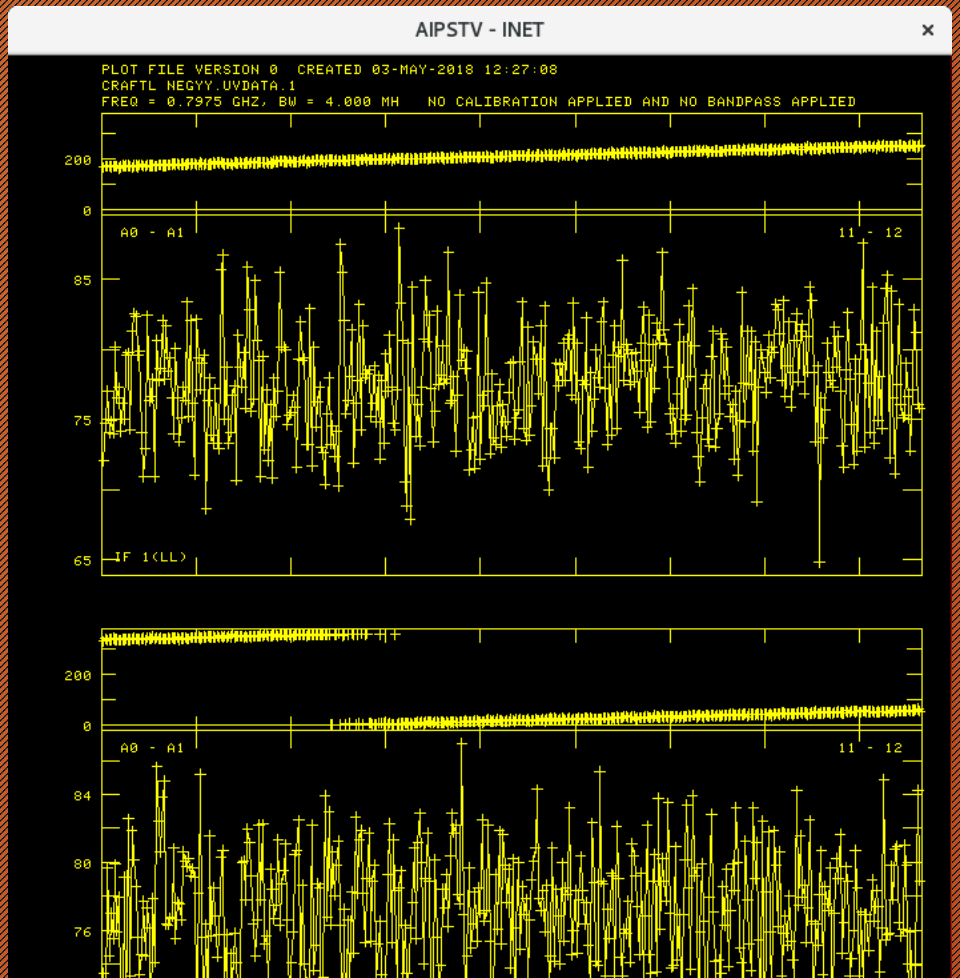
```
adeller@ar313-adeller trunk SB1640> cat run.sh
#!/bin/sh

rm -rf craftfrb_1.difx
rm -rf log*
errormon2 6 &
export ERRORMONPID=$!
mpirun -machinefile machines -np 6 mpifxcorr craftfrb_1.input
kill $ERRORMONPID
rm -f craftfrb_1.difxlog
mv log craftfrb_1.difxlog
```

```
adeller@ar313-adeller trunk SB1640> cat craftfrb_1.stitchconfig
[config]
target_bw: 4.000
target_nchan: 432
target_chavg: 1
stitch_oversamplenum: 32
stitch_oversampledenom: 27
stitch_nstokes: 1
stitch_antennas: *
stitch_basefreqs: 1169.5, 1173.5
verbose: True
```

ASKAP results: pre- and post-stitching





Status in DiFX

- CODIF branch was branched >1 year ago
- Now fairly well tested; should merge back into trunk asap before too much more divergence
  - Lower sideband complex currently has an error (should be conjugated)
- Realistic to do that merging this week? (I think yes)
- But we would want to run tests post-merge on both complex and non-complex data (to make sure we didn't break anything...)
  - Developers willing to run some regression tests?