DiFX1.5 File Formats

There's 5 main ascii files you need to run a correlation (6 if you're doing pulsar binning) - only one of them is really very complex. I'll start with the easiest and work my way up. Remember, check out the examples for more info, or these examples for some much more complicated setups including pulsar binning. Whenever a keyword/value pair is referred to, the value begins at the 21st character. Also, sorry the tabs don't come out properly in the example delay and uvw file snippets on this page.

The machine file

Simple - 1 line per node of the correlation. If you request more nodes than this file has lines, mpi will wrap back to the start - not efficient. An example for a 10 node correlation on the Swinburne cluster might be:

```
tera01
tera02
tera03
tera04
tera05
tera06
tera07
tera08
tera09
tera10
```

The threads file

Just as simple as the machine file, the threads file details how many threads you want for each node that will be a Core. So if there's 10 nodes, and this is 3 station experiment, there will be 10 - 3 (datastreams) - 1 (manager) = 6 Core nodes. That means your threads file should be at least 6 lines long. It starts with one line telling how many Cores there can be, and then has Ncores lines with just a number per line. That number is how many threads for that node. So it looks something like:

```
NUMBER OF CORES: 6
2
2
2
2
2
2
```

In this example, tera05, tera06,...tera10 have 2 threads each. This is sensible when you have a dual-core machine, or one with hyperthreading. Being able to specify the threads on a per-node basis lets you squeeze the best performance out of a heterogenous cluster.

The uvw file

This has the precomputed uvw values (in metres) for each telescope in the correlation. It starts off with some keyword/value information on the telescopes and the timerange covered, and then lists the uvw values at a number of points in each scan. In more detail: 6 keyword value lines for the date and time eg:

```
START YEAR:         2007
START MONTH:        1
START DAY:          27
START HOUR:         0
START MINUTE:       59
START SECOND:       0.0
```

A keyword/value pair indicating the time in seconds between uvw measurements in the file eg:

```
INCREMENT (SECS):   1
```

A table of the antennas in the file, with their location, mount and axis offset, eg:

```
NUM TELESCOPES:     3
TELESCOPE 0 NAME:   PKS
TELESCOPE 0 MOUNT:  azel
TELESCOPE 0 X (m):  -4554231.656
TELESCOPE 0 Y (m):  2816759.097
TELESCOPE 0 Z (m):  -3454036.085
TELESCOPE 1 NAME:   CATW172
TELESCOPE 1 MOUNT:  azel
TELESCOPE 1 X (m):  -4751111.121
TELESCOPE 1 Y (m):  2792597.147
TELESCOPE 1 Z (m):  -3200491.7
TELESCOPE 2 NAME:   MOPRA
TELESCOPE 2 MOUNT:  azel
TELESCOPE 2 X (m):  -4682768.63
TELESCOPE 2 Y (m):  2802619.06
TELESCOPE 2 Z (m):  -3291759.9
```

A keyword/value pair for the number of scans in the file eg:

```
NUM SCANS:          442
```

Then, for each scan, we have the a short header detailing the start point (in units of the `increment' specified above), number of points in this scan, and source information:

```
SCAN 0 POINTS:      3660
SCAN 0 START PT:    0
SCAN 0 SRC NAME:    2255-282
SCAN 0 SRC RA:      6.01309290672888
SCAN 0 SRC DEC:     -0.488213469533334
```

This is followed by N lines of uvw information, where N is three more than the number of points in the scan. The 3 extra points are 1 before the start of the scan, the end of the scan, and one after the end of the scan. They are necessary to allow quadratic interpolation of the uvws in DiFX.
The lines themselves consist of 3 double precision values per telescope (u,v,w) separated by tabs:

```
RELATIVE INC -1:    -4422923.400427 -1635977.07993768 4285656.48881794
-4480653.93533355 -1323071.2766 4431 4334515.70411182 -4462108.79816306
-1434876.53542624 4318494.43016203 -3908986.436378 -2596355.30353404
4305222.82165622 -4262266.22917909 -1845427.2806343 4361869.43587185
```

In this example, PKS u=-4422923.400427m, v=-1635977.07993768m, w=4285656.48881794m, CATW172 u=-4480653.93533355m ...
The next scan information follows immediately after the last line of the previous scan:

```
RELATIVE INC 3661:  -3469314.08270723 -1139659.67209665 5222122.07593112
-3474925.64487348 -822771.36799 4098 5278517.03745916 -3474594.52514974
-935763.342832722 5260248.94742375 -3087756.61648574 -2156383.35833915
5135362.56754033 -3322476.95372888 -1368461.96043286 5261832.2277425
SCAN 1 POINTS:      720
SCAN 1 START PT:    3660
...
```

The delay file

This has the precomputed geocentric delay values for each telescope, and is very similar in format to the uvw file. The first 7 lines are identical (start date/time and time increment in seconds), and are followed by a telescope table, as with the uvw file. However, in the delay file, only the telescope's names are given (rather than name, xyz and mount) ie:

```
NUM TELESCOPES:     3
TELESCOPE 0 NAME:   PKS
TELESCOPE 1 NAME:   CATW172
TELESCOPE 2 NAME:   MOPRA
```

The same NUMBER OF SCANS line follows, and then the actual delay information. The scan header is identical to the uvw scan header except with the RA and DEC information removed, ie:

```
SCAN 0 POINTS:      3660
SCAN 0 START PT:    0
SCAN 0 SRC NAME:    2255-282
```

As with the uvw file, there are 3 extra lines per scan of delay info. Each line of the scan block consists of one double precision number per telescope, tab separated, which is the geometric delay in microseconds for that telescope:

```
RELATIVE INC -1:    14296.7612598859 14459.6988020343 14406.2729168216
14362.0826380644 14550.9768424 732
```

In this example, the geometric delay for PKS is 14296.7612598859, CATW172 is 14459.6988020343, …

The correlator input file

This is of necessity a fairly complex file, and fairly long, although typically a lot of it is just repetition for different baselines and telescopes, and easy to generate automatically from the vex file. It is divided into a series of tables, which I will go through in turn.

The common settings table

This contains general information such as time range, and the paths to the other ascii files described above. The necessary keywords are shown below, with notes if the meaning is not obvious:

```
DELAY FILENAME:     {the delay file path}
UVW FILENAME:       {the uvw file path}
CORE CONF FILENAME: {the thread file path}
EXECUTE TIME (SEC):
START MJD:
START SECONDS:      {offset from 00:00:00 on the start day}
ACTIVE DATASTREAMS:
ACTIVE BASELINES:   {usually Ndatastreams*(Ndatastreams-1)/2, unless some
aren't worth correlating}
DATA HEADER O/RIDE: {Always false. Provided for future compatibility when
baseband files have more metadata}
OUTPUT FORMAT:      {Currently, available options are RPFITS, ASCII and
SWIN}
OUTPUT FILENAME:    {the rpfits filename if FORMAT=RPFITS, or the directory
for output files if FORMAT=SWIN}
```

The config table

This contains info on correlator setup - number of channels per IF, message sizes etc. This is placed in a separate table to the common settings so that you can have different setups for different sources - ie high frequency resolution for a target maser and low frequency resolution for your continuum phase reference source. It also allows you to turn pulsar binning on for specific sources. The first line just informs us how many configs will follow:

```
NUM CONFIGURATIONS: 6
```

So then we get one of the following blocks of information per config - I've shown some typical values:

```
CONFIG SOURCE:      1604-4441 {The source name}
INT TIME (SEC):     2.0 {Integration time in seconds}
NUM CHANNELS:       64 {Spectral points per subbands}
BLOCKS PER SEND:    1000 {Number of FFT blocks to be processed at a time by
```

```
a Core}
GUARD BLOCKS:       1 {Extra FFT blocks to tack onto the end of a message}
POST-F FRINGE ROT:  FALSE {Whether to do fringe-rotation using a constant
approximation across an FFT length}
QUAD DELAY INTERP:  TRUE {Whether to do fringe-rotation using a quadratic
approximation across an FFT length}
WRITE AUTOCORRS:    TRUE {Whether to write autocorrelations to disk}
PULSAR BINNING:     FALSE {Does this source need pulsar binning?}
DATASTREAM 0 INDEX: 0 {Refers to the Datastream table, explained below}
DATASTREAM 1 INDEX: 1 {There are 5 since we specified 5 Datastreams in the
Common Settings}
DATASTREAM 2 INDEX: 2
DATASTREAM 3 INDEX: 3
DATASTREAM 4 INDEX: 4
BASELINE 0 INDEX:   0 {Refers to the Baseline table, explained below}
BASELINE 1 INDEX:   1 {There are 10 since we specified 10 Baselines in the
Common Settings}
BASELINE 2 INDEX:   2
BASELINE 3 INDEX:   3
BASELINE 4 INDEX:   4
BASELINE 5 INDEX:   5
BASELINE 6 INDEX:   6
BASELINE 7 INDEX:   7
BASELINE 8 INDEX:   8
BASELINE 9 INDEX:   9
```

Usually, at least one of the config sources will be DEFAULT. This is the config used for any source without an individual specification. If no DEFAULT config is specified, then sources which do not appear in the CONFIG table are skipped over and not correlated.

If PULSAR BINNING is TRUE, an extra line is inserted immediately below the PULSAR BINNING line as shown below:
PULSAR CONFIG FILE:
/nfs/cluster/ska/adeller/v190/v190f/pulseprofiles/2144-3933/2144-3933.gate.binconfig
The format of the pulsar config file is described below.

The frequency table

Lists all the frequencies used in the experiment. Like most of these tables, it starts with one line listing the number of entries, and then has three lines per entry: band edge frequency, upper or lower sideband, and the bandwidth. The frequencies are specified in MHz, and U or L is used to indicate upper/lower sideband respectively. A sample freq table is shown below:

```
FREQ ENTRIES:       4
FREQ (MHZ) 0:       1634.0
BW (MHZ) 0:         16.0
SIDEBAND 0:         L
FREQ (MHZ) 1:       1634.0
BW (MHZ) 1:         16.0
SIDEBAND 1:         U
```

```
FREQ (MHZ) 2:       1666.0
BW (MHZ) 2:         16.0
SIDEBAND 2:         L
FREQ (MHZ) 3:       1666.0
BW (MHZ) 3:         16.0
SIDEBAND 3:         U
```

All future tables refer to the freq table when specifying frequency bands.

The telescope table

The telescope table contains a listing of the stations used in the experiment. The names used must be a subset of those in the delay and uvw files - the correlator will die gracefully if it cannot find one of the stations in this table somewhere in the delay and uvw files. Each station has a clock offset (microseconds) and a clock rate (microseconds per second). These are in the same sense as the geometric delay ie a positive clock offset is a *delay*. Thus, if you are looking at the delay quantity of an SN table in AIPS, the corrections you make to these numbers are in the same sense as those you see on the TV. An example telescope table is shown below.

```
# TELESCOPE TABLE ##!
TELESCOPE ENTRIES:  5
TELESCOPE NAME 0:   PKS
CLOCK DELAY (us) 0: 0.0
CLOCK RATE(us/s) 0: 0.0
TELESCOPE NAME 1:   CATW172
CLOCK DELAY (us) 1: -49.14
CLOCK RATE(us/s) 1: 6.94E-8
TELESCOPE NAME 2:   MOPRA
CLOCK DELAY (us) 2: -3.455
CLOCK RATE(us/s) 2: -1.2199E-6
...
```

Entries in the telescope table are referred to by the Datastream table entries. Thus, more than one Datastream can reference a single Telescope. This is arranged in this fashion so you don't need to specify the station clocks over and over again, when you have a few different band setups throughout the experiment (ie wideband phase reference, narrowband target etc). It is also useful if one station has recorded separate streams of data - this happens at the LBA in 1 Gbps mode, where the data is recorded in two separate 512 Mbps files. In this situation, you really have two “Datastreams” coming from one “Telescope”.

The datastream table

The table starts with the usual number of entries, and then two lines which affect all Datastreams. These are the factors affecting the size and breakup of the memory buffer. The size of the buffer is given in terms of a multiplier for the message size (which is itself a number of FFT chunks - see the Config table). The memory buffer is then divided into a number of segments - this must be even and must be at least 4.

```
# DATASTREAM TABLE #!
DATASTREAM ENTRIES: 5
DATA BUFFER FACTOR: 32
NUM DATA SEGMENTS:  8
```

The table entries are necessarily complex, as they completely describe the band setup for each datastream. This comprises the format and precision of the recording, the a priori system temperature, the data source (network or disk), whether to use a filterbank instead of an FFT, the number of frequencies, small delay offsets for each frequency, the number of polarisations recorded in each frequency and finally the order of each of the bands within the file.

The introductory stuff (format, tsys etc) goes at the top as shown:

```
TELESCOPE INDEX:    0
TSYS:               42.0
DATA FORMAT:        LBAVSOP
QUANTISATION BITS:  2
FILTERBANK USED:    FALSE
READ FROM FILE:     TRUE
```

Choices for the Mode include LBA (2 bit mag sign encoding), LBAVSOP (2 bit offset binary encoding), MKV, and NZ (8 bit linear). If MKV format is used, an additional line is inserted between the DATA FORMAT and the QUANTISATION BITS to tell the correlator the fanout of the data, eg:

```
FANOUT:             2
```

This is followed by a frequency section, which lists the number of frequencies, indexes to the frequency table, small delay offsets for each frequency (usually 0 - if used, applied in the same sense as AIPS displays residual delays), and the number of polarisations for each frequency (1 or 2 obviously):

```
NUM FREQS:          4
FREQ TABLE INDEX 0: 0
CLK OFFSET 0 (us):  0.01
NUM POLS 0:         2
FREQ TABLE INDEX 1: 1
CLK OFFSET 1 (us):  0.01
NUM POLS 1:         2
FREQ TABLE INDEX 2: 2
CLK OFFSET 2 (us):  0.0
NUM POLS 2:         2
FREQ TABLE INDEX 3: 3
CLK OFFSET 3 (us):  0.0
NUM POLS 3:         2
```

Now, if you add up the polarisations from each frequency, in the example above it is clear there are 8 bands total for this datastream. So, the final part of the entry for this datastream is 8 band entries, each with a frequency (an index to the “local frequency table” - ie the section from just above), and a polarisation:

```
INPUT BAND 0 POL:   L
INPUT BAND 0 INDEX: 0
INPUT BAND 1 POL:   L
INPUT BAND 1 INDEX: 1
INPUT BAND 2 POL:   R
INPUT BAND 2 INDEX: 0
INPUT BAND 3 POL:   R
INPUT BAND 3 INDEX: 1
INPUT BAND 4 POL:   L
INPUT BAND 4 INDEX: 2
INPUT BAND 5 POL:   L
INPUT BAND 5 INDEX: 3
INPUT BAND 6 POL:   R
INPUT BAND 6 INDEX: 2
INPUT BAND 7 POL:   R
INPUT BAND 7 INDEX: 3
```

So to work out the actual frequency for a given band takes two lookups: say band 6 in the example above, that references “local frequency” 2. We look at FREQ TABLE INDEX 2 from the local frequency table: that references entry 2 of the actual frequency table. Looking at that, we see that it is sky frequency 1666 MHz, lower side band, with a bandwidth of 16 MHz.
If all telescopes are configured identically then the “local frequency table” is degenerate with the actual frequency table and hence pretty boring - its when you have multiple setups and telescopes with different recording modes that it gets useful. Its really a convenience thing for the way the correlator looks stuff up internally anyway.

For use in the baseline section, I'll also show the second Datastream (CATW172):

```
TELESCOPE INDEX:    1
TSYS:               68.0
DATA FORMAT:        LBAVSOP
QUANTISATION BITS:  2
FILTERBANK USED:    FALSE
READ FROM FILE:     TRUE
NUM FREQS:          4
FREQ TABLE INDEX 0: 0
CLK OFFSET 0 (us):  0.0
NUM POLS 0:         2
FREQ TABLE INDEX 1: 1
CLK OFFSET 1 (us):  0.0
NUM POLS 1:         2
FREQ TABLE INDEX 2: 2
CLK OFFSET 2 (us):  0.0
NUM POLS 2:         2
FREQ TABLE INDEX 3: 3
CLK OFFSET 3 (us):  0.0
NUM POLS 3:         2
INPUT BAND 0 POL:   R
INPUT BAND 0 INDEX: 0
INPUT BAND 1 POL:   R
INPUT BAND 1 INDEX: 1
```

```
INPUT BAND 2 POL:   L
INPUT BAND 2 INDEX: 0
INPUT BAND 3 POL:   L
INPUT BAND 3 INDEX: 1
INPUT BAND 4 POL:   R
INPUT BAND 4 INDEX: 2
INPUT BAND 5 POL:   R
INPUT BAND 5 INDEX: 3
INPUT BAND 6 POL:   L
INPUT BAND 6 INDEX: 2
INPUT BAND 7 POL:   L
INPUT BAND 7 INDEX: 3
```

The baseline table

The baseline table starts with the usual “number of entries” line.

```
# BASELINE TABLE ###!
BASELINE ENTRIES:   10
```

Each entry then consists of two Datastreams (references to the Datastream table), the number of frequencies, and the number of polarisation products per frequency, as shown below:

```
D/STREAM A INDEX 0: 0
D/STREAM B INDEX 0: 1
NUM FREQS 0:        4
POL PRODUCTS 0/0:   2
D/STREAM A BAND 0:  2
D/STREAM B BAND 0:  0
D/STREAM A BAND 1:  0
D/STREAM B BAND 1:  2
POL PRODUCTS 0/1:   2
D/STREAM A BAND 0:  3
D/STREAM B BAND 0:  1
D/STREAM A BAND 1:  1
D/STREAM B BAND 1:  3
POL PRODUCTS 0/2:   2
D/STREAM A BAND 0:  6
D/STREAM B BAND 0:  4
D/STREAM A BAND 1:  4
D/STREAM B BAND 1:  6
POL PRODUCTS 0/3:   2
D/STREAM A BAND 0:  7
D/STREAM B BAND 0:  5
D/STREAM A BAND 1:  5
D/STREAM B BAND 1:  7
```

If we look up the Datastream table, we see that Datastream 0 and 1 reference telescope 0 and 1, which are PKS and CATW172 respectively. Each of these Datastreams has 4 frequencies, so it is

unsurprising that we are choosing to correlate all 4. Each frequency here has two polarisation products, and if we again follow the references back through the Datastream table, we see that in each case the products correspond to RR and LL. Eg for the first frequency, band 2 of PKS is 1634 LSB, polarisation R, and band 0 of CATW172 is 1634 LSB, polarisation R, so this product is 1634 RR. Band 0 of PKS is 1634 LSB, polarisation L, and band 2 of CATW172 is 1634 LSB, polarisation L, so this product is 1634 LL. Naturally, most of the time you would use a script to set this table up, looking for all bands that overlap for the given baseline.

The data table

This table must be included if one or more datastreams read from a file. It is implicitly the same length as the datastream table (there is no “number of entries” line). Each datastream has one line to say the number of files N, and then N lines with filenames:

```
# DATA TABLE #######!
D/STREAM 0 FILES:   8639
FILE 0/0:           /nfs/cluster/raid9/v190f/v190f-Pk_027_020000.lba
FILE 0/1:           /nfs/cluster/raid9/v190f/v190f-Pk_027_020010.lba
...
```

The network table

This table must be included if one or more datastreams read from a network connection (READ FROM FILE: FALSE). It is implicitly the same length as the datastream table (there is no “number of entries” line). Each datastream has two lines - a port number and a TCP window size in kB.

```
# NETWORK TABLE ####!
PORT NUM 0:         10001
TCP WINDOW SIZE 0:  250
PORT NUM 1:         10002
TCP WINDOW SIZE 1:  250
...
```

Probably best to contact me if you have interest in trying out the network-fed correlator, as you'll need to set up the sending side of things as well which isn't covered here.

The pulsar configuration file

This is pretty simple - it gives links to the polyco file(s) containing pulse prediction information (see the program TEMPO for a description of the polyco file format), and specifies where the bin end-points are set. It also gives the option to “scrunch” the binned data. If SCRUNCH is true, each bin is scaled by its corresponding weight and the bins are summed before writing to disk: thus only one “bin” is recorded per time integration. This can be used to implement a matched filter for each pulsar, recovering maximum S/N. If SCRUNCH is false, each bin is written out separately and the weights are ignored. This mode is not well tested, and may have bugs.

```
NUM POLYCO FILES:   3
POLYCO FILE 0:
/nfs/cluster/ska/adeller/v190/v190f/pulseprofiles/0630-2834/0630-2834_54126_
200000.polyco
POLYCO FILE 1:
/nfs/cluster/ska/adeller/v190/v190f/pulseprofiles/0630-2834/0630-2834_54127_
120000.polyco
POLYCO FILE 2:
/nfs/cluster/ska/adeller/v190/v190f/pulseprofiles/0630-2834/0630-2834_54127_
200000.polyco
NUM PULSAR BINS:    2
SCRUNCH OUTPUT:     TRUE
BIN PHASE END 0:    0.58
BIN WEIGHT 0:       0.0
BIN PHASE END 1:    0.665
BIN WEIGHT 1:       1.0
```

This example shows a simple gate, where only data falling between pulse phase 0.58 and 0.665 is retained.

The SWIN output data format

Okay, so this isn't an ascii control file, but it is a file format so I'll describe it briefly here. The purpose of this file is to hold a bunch of visibilities in a relatively easy to understand format, which you can then translate into your favourite flavour of FITS or similar. At Swinburne we're working on AIPS++ measurement sets right now.

You create “SWIN” style output data by specifying OUTPUT FORMAT: SWIN in the common table of your correlator input file. When creating SWIN style data, the OUTPUT FILENAME keyword in the common table must refer to a non-existent directory that you want to create to store the visbility files in. The root directory of the directory you specify must exist eg if you want to use /tmp/experiment/binary/ as your output directory, /tmp/experiment/ must exist but /tmp/experiment/binary/ must not.

In this directory, one or more SWIN style visibility files will be created. Each file will have a name of the form

DIFX_MJD_SECONDS.nnnnn

where MJD is the MJD of the first visibility point in the file, SECONDS is the number of seconds since the start of MJD for the first visibility point, and nnnnn is the number of visibility entries in the file.

Each visibility entry consists of a short ascii header (containing the kind of info that is held in the random group headers of RPFITS), followed by the visibility data in 32 bit complex floats, and optionally weights as 32 bit complex floats. Each header has 13 keyword/value pairts and looks like this:

```
BASELINE NUM:       258
MJD:                54044
```

```
SECONDS:            3600.5
CONFIG INDEX:       0
SOURCE INDEX:       1
FREQ INDEX:         0
POLARISATION PAIR:  RR
PULSAR BIN:         0
FLAGGED:            0
DATA WEIGHT:        1.0
U (METRES):         -4422923.400427
V (METRES):         -1635977.07993768
W (METRES):         4285656.48881794
```

The header is immediately followed by the binary real and imag for each point. The length will be 2*numchannels floats, packed as re im re im re ...
The value numchannels can be found from the input file, looking at the correct entry in the config table as specified by CONFIG INDEX. The end of the visibilities is immediately followed by the next header, and so on.