Control File Formats for DiFX

There was a major break in the control file formats between DiFX1.5 and DiFX2.0. If you want the formats for DiFX1.5, please click through to difx1.5-files. This page has the file formats for DiFX2.0. Now, you shouldn't really need to look as these at all if you are using vex2difx as recommended. If you a building a 3rd party application, consider using the difxio library (see SVN for details) - it parses all this info in c structures, saving you the hassle.

There's 5 main ascii files you need to run a correlation (plus extras if you're doing pulsar binning or phased array mode) - 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 (or after the : separator if the keyword is longer than 20 characters). Also, sorry the tabs don't come out properly in the example .im 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 MPI processes 2-N+1 (N==#telescopes) will be datastream processes.

Say you have host1 and host2. host1 is “head” and all the data is stored there and you have 4 telescopes. You want to run a core process on both nodes.

Then the number of mpi processes will be 7 (1+4+2 == fxmanager + 4xdatastream + 2xcore).

Use the machines files to get the processes in the right place. This depends a little on your implementation of mpi. From memory the following should work: for my example above. Don't include the comments!

```
  □host1        # Fxmanager
  □host1        # datastream 1
  □host1        # datastream 2
  □host1
  □host1        # datastream 4
 host1        # core
 host2        # core
```

Remember that “espresso” and such tools handle this for you.

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.

If writing this file by hand be careful that the value for number of cores *must* start in column 21 (the same as most of the values in the other input files). Getting this wrong will cause all core processes to run with a single thread, and could even cause mpifxcorr to crash.

The .calc file

The .calc file stores all the information on antenna locations, source locations, and scans (start times, durations etc). It also has Earth Orientation Parameters necessary to run CALC. The calc file is supplied to calcif2, which produces the .im file described below. An example .calc file follows (“…” shows lines in a table following a pattern which is hopefully already clear):

```
JOB ID:             20
JOB START TIME:     53440.9226852
JOB STOP TIME:      53440.9228009
DUTY CYCLE:         1.000
OBSCODE:            V177A
DIFX VERSION:       DiFX-2.0
SUBJOB ID:          0
SUBARRAY ID:        0
START MJD:          53440.9226852
START YEAR:         2005
START MONTH:        3
START DAY:          11
START HOUR:         22
START MINUTE:       8
START SECOND:       40
SPECTRAL AVG:       1
TAPER FUNCTION:     UNIFORM
NUM TELESCOPES:     4
TELESCOPE 0 NAME:   AT
TELESCOPE 0 MOUNT:  azel
TELESCOPE 0 OFFSET (m):0.000000
TELESCOPE 0 X (m):  -4751685.988000
TELESCOPE 0 Y (m):  2791621.223000
TELESCOPE 0 Z (m):  -3200491.700000
TELESCOPE 0 SHELF:  NONE
TELESCOPE 1 NAME:   HO
...
TELESCOPE 3 SHELF:  NONE
NUM SOURCES:        1
SOURCE 0 NAME:      1600-445
SOURCE 0 RA:        4.2084997996221
SOURCE 0 DEC:       -0.7800257309396
SOURCE 0 CALCODE:
SOURCE 0 QUAL:      0
NUM SCANS:          1
SCAN 0 IDENTIFIER:  No0024
SCAN 0 START (S):   0
SCAN 0 DUR (S):     10
SCAN 0 OBS MODE NAME:Doppler@G329+0.6
SCAN 0 UVSHIFT INTERVAL (NS):2000000000
SCAN 0 AC AVG INTERVAL (NS):2000000000
SCAN 0 POINTING SRC:0
SCAN 0 NUM PHS CTRS:1
SCAN 0 PHS CTR 0:   0
NUM EOPS:           5
EOP 0 TIME (mjd):   53438
EOP 0 TAI_UTC (sec):33
EOP 0 UT1_UTC (sec):-0.443500
EOP 0 XPOLE (arcsec):0.006740
EOP 0 YPOLE (arcsec):0.218860
EOP 1 TIME (mjd):   53439
...
EOP 4 YPOLE (arcsec):0.219710
NUM SPACECRAFT:     0
IM FILENAME:        /users/adeller/testing/2.0/noshift/lba_20.im
```

The .im file

DiFX1.5 specified the geometric model using two files - the .delay and .uvw files. These held sampled values of the delay and uvw values (from CALC) on a regular grid - typically one sample per second, for every antenna. The .im file supersedes the .delay and .uvw files, and instead of sampling every second it instead stores a polynomial less frequently, typically once per two minutes.

An example .im file follows (same rule for “…”):

```
CALC SERVER:        swc000
CALC PROGRAM:       536871744
CALC VERSION:       1
START YEAR:         2005
START MONTH:        3
START DAY:          11
START HOUR:         22
START MINUTE:       8
START SECOND:       40
POLYNOMIAL ORDER:   5
INTERVAL (SECS):    120
ABERRATION CORR:    EXACT
NUM TELESCOPES:     4
TELESCOPE 0 NAME:   AT
TELESCOPE 1 NAME:   HO
TELESCOPE 2 NAME:   MP
TELESCOPE 3 NAME:   PA
NUM SCANS:          1
SCAN 0 POINTING SRC:1600-445
SCAN 0 NUM PHS CTRS:1
SCAN 0 PHS CTR 0 SRC:1600-445
SCAN 0 NUM POLY:    2
SCAN 0 POLY 0 MJD:  53440
SCAN 0 POLY 0 SEC:  79680
SRC 0 ANT 0 DELAY (us): 1.590699551807348e+04   -7.299033971692804e-01
-2.232549659111185e-05     6.468692358170586e-10     9.946319030146373e-15
-3.769275660019511e-19
SRC 0 ANT 0 DRY (us): 9.974292374186537e-03  4.541839076455105e-07
3.452769928524840e-11     1.795348217396608e-15     1.046710943091233e-19
5.943899753099940e-24
SRC 0 ANT 0 WET (us): 4.656132867051395e-04  2.125435426767988e-08
1.619354861844071e-12     8.461244831771577e-17     4.959121378098075e-21
2.839532897018426e-25
SRC 0 ANT 0 U (m):  -4.222122069441900e+06  -2.581071707998739e+02
1.122949738068942e-02     2.223645238970012e-07     5.617088869943636e-11
-2.068303157154808e-13
SRC 0 ANT 0 V (m):  -2.141630031993518e+05   2.167117333885627e+02
6.616688709965712e-03    -1.854345842529444e-07    -6.531226953877621e-11
2.068221832176051e-13
SRC 0 ANT 0 W (m):  -4.768800415563213e+06   2.188193910068975e+02
6.693004677799763e-03    -1.939274091983166e-07    -2.979016210693149e-12
1.042909093213190e-16
SRC 0 ANT 1 DELAY (us): 1.756912593080410e+04   -6.010048711065136e-01
-1.981287933637709e-05     5.326361025763266e-10     8.818734493518715e-15
-2.969732343177518e-19
...
```

```
SRC 0 ANT 3 W (m):  -4.941474293258900e+06   2.085116694419438e+02
6.677883811385578e-03    -1.847904513851888e-07    -2.988139228644093e-12
1.530249792227679e-16
SRC 1 ANT 0 DELAY (us): 1.590699551807348e+04   -7.299033971692804e-01
-2.232549659111185e-05     6.468692358170586e-10     9.946319030146373e-15
-3.769275660019511e-19
SRC 1 ANT 0 DRY (us): 9.974292374186537e-03  4.541839076455105e-07
3.452769928524840e-11     1.795348217396608e-15     1.046710943091233e-19
5.943899753099940e-24
SRC 1 ANT 0 WET (us): 4.656132867051395e-04  2.125435426767988e-08
1.619354861844071e-12     8.461244831771577e-17     4.959121378098075e-21
2.839532897018426e-25
SRC 1 ANT 0 U (m):  -4.222122069441900e+06  -2.581071707998739e+02
1.122949738068942e-02     2.223645238970012e-07     5.617088869943636e-11
-2.068303157154808e-13
SRC 1 ANT 0 V (m):  -2.141630031993518e+05   2.167117333885627e+02
6.616688709965712e-03    -1.854345842529444e-07    -6.531226953877621e-11
2.068221832176051e-13
SRC 1 ANT 0 W (m):  -4.768800415563213e+06   2.188193910068975e+02
6.693004677799763e-03    -1.939274091983166e-07    -2.979016210693149e-12
1.042909093213190e-16
SRC 1 ANT 1 DELAY (us): 1.756912593080410e+04   -6.010048711065136e-01
-1.981287933637709e-05     5.326361025763266e-10     8.818734493518715e-15
-2.969732343177518e-19
...
SRC 1 ANT 3 V (m):  -2.846209623790657e+04   2.065102986336044e+02
6.602685843807587e-03    -1.893219703175936e-07     4.310608957123050e-11
-1.197391361113666e-13
SRC 1 ANT 3 W (m):  -4.941474293258900e+06   2.085116694419438e+02
6.677883811385578e-03    -1.847904513851888e-07    -2.988139228644093e-12
1.530249792227679e-16
SCAN 0 POLY 1 MJD:  53440
SCAN 0 POLY 1 SEC:  79800
SRC 0 ANT 0 DELAY (us): 1.581908674310539e+04   -7.352335032343110e-01
-2.209177040755237e-05     6.515979361014590e-10     9.781384664758747e-15
-1.313535485117763e-19
...
```

Note that “SRC 0” is always the pointing centre source, and SRC 1 through to SRC N are the N phase centres. The rows contain the polynomial coeffients, running left to right as you'd expect.

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 (using vex2difx). It is divided into a series of tables, which I will go through in turn. Lines beginning with # denote the start of a table, and lines beginning with @ are comments which are ignored.

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:

```
CALC FILENAME:      /users/adeller/testing/2.0/noshift/lba_20.calc    {The
path to the calc file}
CORE CONF FILENAME: /users/adeller/testing/2.0/noshift/lba_20.threads {The
path to the threads file}
EXECUTE TIME (SEC): 10
START MJD:          53440
START SECONDS:      79720
ACTIVE DATASTREAMS: 4
ACTIVE BASELINES:   6
VIS BUFFER LENGTH:  32                                                {A
buffer length at the FxManager}
OUTPUT FORMAT:      SWIN                                              {Must
be SWIN to use difx2fits; ASCII is available for debugging}
OUTPUT FILENAME:    /users/adeller/testing/2.0/noshift/lba_20.difx    {The
directory where output files will be written}
```

The config table

This contains info on correlator setup - integration times, 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: 1
```

Then we get info for each of these configurations:

```
CONFIG NAME:        Doppler@G329+0.6_default
INT TIME (SEC):     1.000000
SUBINT NANOSECONDS: 80000000                {Determines the message sizes
from Datastream to Core}
GUARD NANOSECONDS:  2000
FRINGE ROTN ORDER:  1                       {Can be 0 for post-F, 1 for
linear, or 2 for piecewise linear quadratic approximation}
ARRAY STRIDE LENGTH:16                      {Used for optimised
trigonometry calculations}
XMAC STRIDE LENGTH: 128                     {Used to ensure output results
can stay in cache}
NUM BUFFERED FFTS:  1                       {Number of FFTs to compute per
station before XMAC'ing.  Also a cache optimisation thing}
WRITE AUTOCORRS:    TRUE
PULSAR BINNING:     FALSE
PHASED ARRAY:       FALSE
```

```
DATASTREAM 0 INDEX: 0
DATASTREAM 1 INDEX: 1
DATASTREAM 2 INDEX: 2
DATASTREAM 3 INDEX: 3
BASELINE 0 INDEX:   0
BASELINE 1 INDEX:   1
BASELINE 2 INDEX:   2
BASELINE 3 INDEX:   3
BASELINE 4 INDEX:   4
BASELINE 5 INDEX:   5
```

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 rule table

The rule tables describes which configuration will be applied at any given time. Usually this filters on scan attributes such as source, but can also be done in a time-based manner (start and stop times). An time for which no configuration matches will not be correlated. If more than one rule matches a given time, they must all refer to the same configuration.

```
NUM RULES:           1
RULE 0 CONFIG NAME: Doppler@G329+0.6_default
```

This example just applies the one configuration to all time - a pretty common occurrence.

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 seven lines per entry: band edge frequency, upper or lower sideband, bandwidth, number of channels, how many of these should be averaged together after correlation, and the oversample and decimation factors. 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
NUM CHANNELS 0:     128
CHANS TO AVG 0:     8
OVERSAMPLE FAC. 0:  1
DECIMATION FAC. 0:  1
FREQ (MHZ) 1:       1634.0
BW (MHZ) 1:         16.0
SIDEBAND 1:         U
```

```
..
DECIMATION FAC. 3:  1
```

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:   AT
@ ***** Clock poly coeff N: has units microsec / sec^N ***** @
CLOCK COEFF 0/0:    -5.504000000000000e01
CLOCK COEFF 0/1:    -1.879490500000000e-08
TELESCOPE NAME 1:   HO
@ ***** Clock poly coeff N: has units microsec / sec^N ***** @
CLOCK COEFF 0/0:    -1.012400000000000e01
CLOCK COEFF 0/1:    6.940000000000000e-08
...
```

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:               0.000000
DATA FORMAT:        LBASTD
QUANTISATION BITS:  2
DATA FRAME SIZE:    40004096
DATA SOURCE:        FILE
FILTERBANK USED:    FALSE
PHASE CAL INT (MHZ):1
NUM RECORDED FREQS: 4
REC FREQ INDEX 0:   0
CLK OFFSET 0 (us):  0.000000
FREQ OFFSET 0 (Hz): 0.000000
NUM REC POLS 0:     2
REC FREQ INDEX 1:   1
...
NUM REC POLS 3:     2
REC BAND 0 POL:     R
REC BAND 0 INDEX:   0
REC BAND 1 POL:     L
REC BAND 1 INDEX:   0
...
REC BAND 7 POL:     L
REC BAND 7 INDEX:   3
NUM ZOOM FREQS:     0
```

If the TSYS value is > 0.0, the correlator will scale the data online to try and produce estimated visibilities in janskys. If it is ⇐ 0.0, the correlator will produce normalised correlation coefficients instead. The latter is the default way and recommended for use with difx2fits.

Choices for the Mode include LBASTD (2 bit mag sign encoding), LBAVSOP (2 bit offset binary encoding), MKIV, VLBA, and NZ (8 bit linear). The data source can be FILE, MODULE or NETWORK (referring to linux files, Mk5 modules, or a network socket - usually for eVLBI). For each recorded frequency, one can apply a small extra instrumental delay if needed, and a frequency offset if the LO was not set correctly, and finally specify how many polarisations were recorded. After describing each of the frequencies, information follows for each subband (a frequency/polarisation combination) which lets you describe how the subbands are ordered. In the example above there were 4 frequencies each with 2 polarisations, so there are 8 subbands in total.

After the recorded frequencies, it is possible to describe “zoom” frequencies and bands which allow the selection of a subset of the spectral channels produced from a recorded band to be correlated. These frequencies must already be described in the Freq table, and must lie wholly within a recorded band and have the same channelisation. If NUM ZOOM FREQS is set to greater than 0, then the zoom band description follows in exactly the same manner as for the recorded bands, e.g. assuming some appropriate entry in the Freq table in position 4:

```
NUM ZOOM FREQS:     1
ZOOM FREQ INDEX 0:  4
NUM ZOOM POLS 0:    2
ZOOM BAND 0 POL:    R
ZOOM BAND 0 INDEX:  0
ZOOM BAND 1 POL:    L
ZOOM BAND 1 INDEX:  0
```

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:  0
D/STREAM B BAND 0:  0
D/STREAM A BAND 1:  1
D/STREAM B BAND 1:  1
POL PRODUCTS 0/1:   2
D/STREAM A BAND 0:  2
D/STREAM B BAND 0:  2
D/STREAM A BAND 1:  3
D/STREAM B BAND 1:  3
POL PRODUCTS 0/2:   2
D/STREAM A BAND 0:  4
D/STREAM B BAND 0:  4
D/STREAM A BAND 1:  5
D/STREAM B BAND 1:  5
POL PRODUCTS 0/3:   2
D/STREAM A BAND 0:  6
D/STREAM B BAND 0:  6
D/STREAM A BAND 1:  7
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 AT and HO 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 0 of AT is 1634 LSB, polarisation R, and band

0 of HO is 1634 LSB, polarisation R, so this product is 1634 RR. Band 1 of PKS is 1634 LSB, polarisation L, and band 1 of CATW172 is 1634 LSB, polarisation L, so this product is 1634 LL. vex2difx sets all this up for you, naturally.

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-At_027_020000.lba
FILE 0/1:           /nfs/cluster/raid9/v190f/v190f-At_027_020010.lba
...
```

When reading from a Mk5 module, there will only be one “file” per datastream, and that will be the module name.

The network table

This table must be included if one or more datastreams read from a network connection (DATA SOURCE: NETWORK). 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. Negative values mean use UDP rather than TCP as the transport protocol.

```
# 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 filename is specified in the input file if PULSAR BINNING is true. If required, it is put on the following line as shown:

```
PULSAR BINNING:     TRUE
PULSAR CONFIG FILE: /home/difx/projects/tc016a/0834+2200.gate.binconfig
```

The format 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 phased array configuration file

This filename is specified in the input file if PHASED ARRAY is true. If required, it is put on the following line as shown:

```
PHASED ARRAY:       TRUE
PHASED ARRAY CONFIG FILE:/home/difx/projects/tc016a/phasedarray.config
```

The format has not been set yet, but will probably look something like this:

```
OUTPUT TYPE:        FILTERBANK [could also be TIMESERIES]
OUTPUT FORMAT:      DIFX [could be VDIF for TIMESERIES]
ACC TIME (NS):      64000 [ignored for TIMESERIES data]
COMPLEX OUTPUT:     TRUE [ignored for FILTERBANK data]
OUTPUT BITS:        8
```

more keys could yet be added - we haven't gotten very far into the implementation yet.

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. difx2fits will produce FITS-IDI data from the SWIN format.

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.s####.b####

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, s#### is the source index (running from 0 to N for a given scan, not the overall source index into the table held in the .calc file) and b#### is the pulsar bin number (again, for the configuration which a given scan refers to).

Each visibility file is completely binary, in comparison to DiFX1.5 which had ascii headers and binary data. The binary header which precedes each visibility entry is of length 74 bytes and contains the following data:

```
Bytes   Type    Contains               Example value
1-4     Int     SYNC WORD              0xFF00FF00
5-8     Int     BINARY HEADER VERSION  1
9-12    Int     BASELINE NUM           258
13-16   Int     MJD                    54044
17-24   Double  SECONDS                3600.5
25-28   Int     CONFIG INDEX           0
29-32   Int     SOURCE INDEX           1
33-36   Int     FREQ INDEX             0
37-38   Char[2] POLARISATION PAIR      RR
39-42   Int     PULSAR BIN             0
43-50   Double  DATA WEIGHT            1.0
51-58   Double  U (METRES)             -4422923.40042
59-66   Double  V (METRES)             -1635977.07993768
67-74   Double  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 endianness of the binary data (header and visibilities) is not enforced, but all instances of DiFX to date use little-endian (Intel format).
In the case of upper sideband data, the first reported channel is the “zero frequency” channel, that is its sky frequency is equal to the value in the frequency table for this spectrum. The Nyquist channel is not retained. For lower sideband data, the last channel is the “zero frequency” channel. That is, in all cases, the spectrum is in order of increasing frequency and the Nyquist channel is excised.
The baseline num is calculated using the 1-based station indices (ie in the example above, AT=1, HO=2…). It is calculated as 256*S1 + S2, where S1 and S2 are the 1-based station indices of the stations that contribute to the baseline.
The CONFIG INDEX, SOURCE INDEX and FREQ INDEX refer to the configuration table, source table and

freq table in the .input, .calc and .input files respectively.
The value numchannels can be found from the input file, looking at the correct entry in the freq table as specified by FREQ INDEX. The end of the visibilities is immediately followed by the next header, and so on.
Because of aliasing, the “numchannels” spectral points that are recorded for each band do not exactly cover the subband bandwidth in the way one might expect. This differs for upper sideband and lower sideband data. For more information, see the channelisation page.

Loop order

For each visibility dump, the individual entries come in the following order:
The top level loop is over baseline, for the NUM ACTIVE BASELINES entries in the current CONFIGURATION (which point at entries in the BASELINE table)
The next level of loop is over frequency, as determined by “NUM FREQS” for each baseline entry.
The next level of loop is phase centre, as determined by the current scan in the .calc file
The next level of loop is over pulsar bin, as determined by the NUM PULSAR BINS for the pulsar bin config for the current CONFIGURATION
The final level of loop is over polarisation, as determined by the “POL PRODUCTS #/#” entry for each baseline frequency.