DiFX 2.x software correlation at IRA
-
An installation and operation manual

Matteo Stagni - Mauro Nanni

September 16, 2013

IRA 471/13

Abstract

DiFX 2 brings a new simplified pipeline for correlating raw data from the antennas. This manual describes the installation and maintenance procedures in the present computing environment at IRA and focuses on how to perform a correlation provided the integrating tools to find fringes. The tasks are presented with best practices to be followed and errors a DiFX operator should avoid. Data format and data streaming are covered extensively due to the recent breakthroughs of dBBC and network streaming tools.

Contents

1 Get DiFX

1.1 Preliminary required components

In order to make DiFX work we need some preliminary packages to be installed. First the Integrated Performance Primitives from Intel, then the message-passing interface OpenMPI for distributing the FFTs on several machines cores and finally the pgplot library which is necessary for plotting correlation results if you are planning to use the Haystack Observatory Postprocessing System (HOPS) [12] package.

1.1.1 IPP

The correlator needs an optimized vector library, which by default (and is currently the only supported option) is Intel Integrated Performance Primitives (IPP) [9]. Intel IPP is a library of highly optimized algorithmic building blocks for media, signal, and data processing applications. Its actual release is 7.1. To install IPP one must run the install script

```
./install.sh
```

found inside the package. Usually the default install directory is something like

```
/opt/intel/composer_xe_2013.1.117
```

though for our distributing needs it is configurable during the install process.

1.1.2 OpenMPI

MPI stands for the Message Passing Interface. Written by the MPI Forum [19] (a large committee comprising of a cross-section between industry and research representatives), MPI is a standardized API typically used for parallel and/or distributed computing. The software can be easily downloaded at open-mpi.org. After configuring a destination install directory through

```
./configure --prefix=/somewhere [--with-openib]
```

you need to run

```
./make install
```

The openib option is to be selected whether an Infiniband connection is available.

1.1.3 Pgplot

Pgplot is a library that is used by HOPS to plot fringes. It is downloadable from Caltech [3], but has a complex install procedure due to the fact that it was developed a few decades ago and is not properly maintained. There are plans to substitute it with plplot library [20]. The following paragraphs are a summary of the installation procedure [4] described on Caltech webpages.

Create a writeable directory in which the PGPLOT library and associated files will be created.

```
mkdir /usr/local/pgplot
```

Configure PGPLOT by selecting device drivers from the available list. First copy the file drivers.list from the distribution directory to the target directory, and then use a text editor to select device drivers. This file contains one line for each available device driver: delete the exclamation mark (!) at the beginning of the line to include the driver, or ensure that an exclamation mark is present if you want to exclude the driver. Many of the drivers can be used only on certain operating systems (see notes in drivers.list), so include only the drivers you plan to use. PGPLOT can later be reconfigured by restarting the installation at this step. Most installations should include: the null device (/NULL), PostScript printers (/PS, /VPS, /CPS, and /VCPS), Tektronix terminals (/TEK, /XTERM, and possibly other variants), and, if the X window system is available on the target, the X window drivers (/XWINDOW, /XSERVE). You may also wish to include drivers for GIF files (/GIF, /VGIF) or some of the other printers.

```
    cd /usr/local/pgplot
    cp /usr/local/src/pgplot/drivers.list .
    vi drivers.list         (or use your preferred editor)
```

The PGPLOT installation procedure for UNIX uses a script, called makemake, to generate a standard UNIX makefile for your operating system, compilers, and list of selected PGPLOT device drivers.

Execute the script makemake from the distribution directory: e.g.

```
/usr/local/src/pgplot/makemake  /usr/local/src/pgplot  linux
```

The script makemake generates a file makefile for subsequent use, a Fortran file grexec.f that calls the selected device drivers, and a text file rgb.txt that contains color definitions for use by routine PGSCRN.

Now use the UNIX make command to compile the PGPLOT library following the instructions in makefile:

```
make
```

If this step proceeds satisfactorily, you may want to type

```
make clean
```

to remove unneeded intermediate files.

1.2 DiFX core svn

DiFX software correlator is a joint project between Swinburne University and NRAO . It comes with a license agreement to be signed for academic purposes usage. In order to be approved the requirement is to join the DiFX-Users group hosted at Google Groups [17]. After receiving an account you will be granted access to the Subversion repository and the Trac website [18]. To get the current version:

```
svn co https://svn.atnf.csiro.au/difx
```

To get a specific version:

```
svn co https://svn.atnf.csiro.au/difx/master_tags/DiFX-2.x
```

DiFX contains five main directories:

```
applications libraries utilities mpifxcorr sites
```

a

```
setup.bash
```

file to source and a key program

```
genipppc
```

which serves the purpose of creating a ipp.pc file in

```
lib/pkgconfig/ipp.pc
```

which determines the right path to include at compilation time for the IPP libraries.

2 Installation on a computing cluster

DiFX design aims primarly at computer clusters. The reason is the high intensive FFT process which requires a good number of cores. Figure 1 describes what mpifxcorr does performing its duty.

DiFX is designed to withstand a single node installation, though performing FFTs on multiple baselines is not recommended.

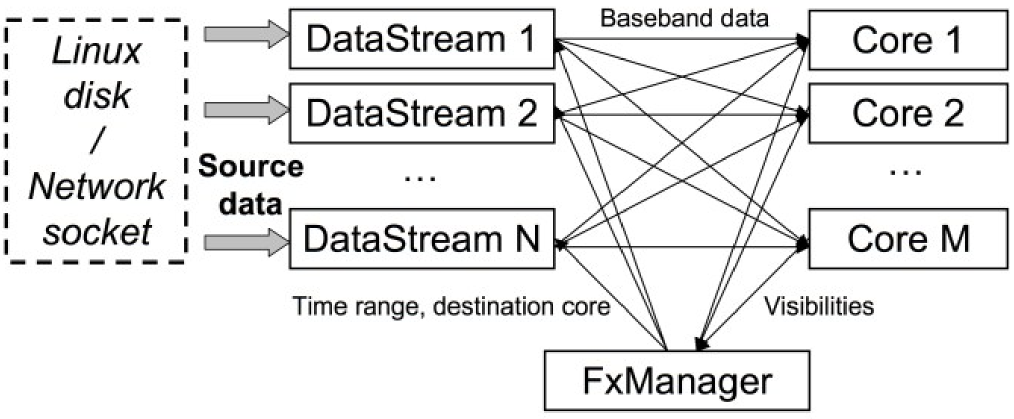


Figure 1: mpifxcorr architecture

2.1 Cluster architecture at IRA

The cluster architecture set up at IRA is made up of a manager (master) node vlbi-mgr that distributes the correlation software to the storage nodes. The storage nodes are interconnected with infiniband through a switch. The design is to provide a single dedicated machine to each antenna of the Italian vlbi network, though in case of need the number of machines assigned for each data storage task can easily grow by adding new machines to the cluster because there are a few steps to take to integrate a new node.

On each machine there is the same dedicated user that is supposed to perform correlation tasks no matter on which of the storage machines one logs in. The correlation user has all the environment setup variables exported and can move around the storage directories to correlate the data on each of them, as the storage space available is exported through nfs on all nodes.

The storage filesystem chosen is XFS because the storage nodes are supposed to record data streams coming from the antennas [15]. The data links for receiving the antennas observations is a 10 Gbit line to the GARR Italian research network.

2.2 Installation example

Taking steps from the previously described architecture, the following sections are about performing a complete installation in such an environment.

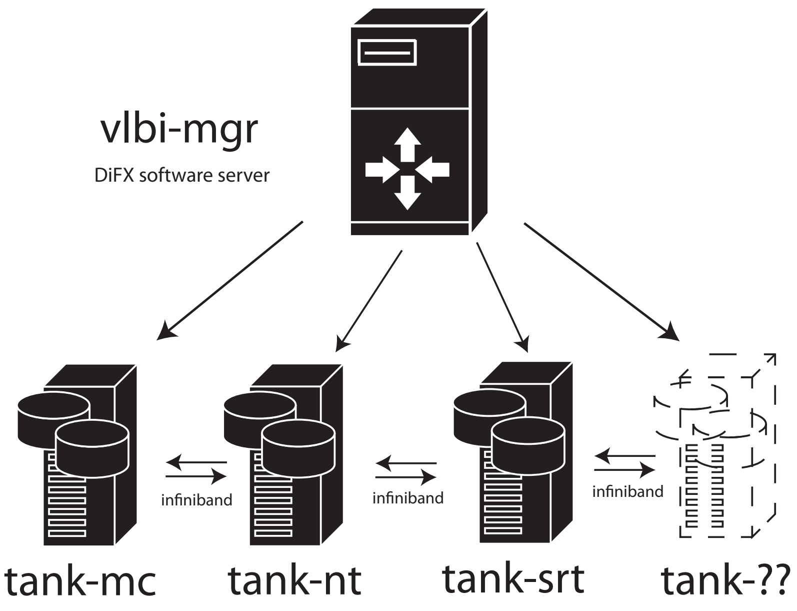


Figure 2: IRA' s cluster architecture

2.2.1 Infiniband setup

The PCIe Infiniband ConnectX-2 VPI cards, model MHQH19B-XTR, from Mellanox were installed on each storage node. We also purchased a 8 port Mellanox InfiniScale IV QDR switch, model MIS5022Q-1BFR.

In order to make the connections work properly, on each node several packages must be installed from the yum repository of Scientific Linux 6.x:

```
ibacm.x86_64 : InfiniBand Communication Manager Assistant
ibibcm.x86_64 : Userspace InfiniBand Connection Manager
libibcommon.x86_64 : OpenFabrics Alliance InfiniBand management common library
libibverbs.x86_64 : A library for direct userspace use of RDMA hardware
libmlx4.x86_64 : Mellanox ConnectX InfiniBand HCA Userspace Driver
libmthca.x86_64 : Mellanox InfiniBand HCA Userspace Driver
mvapich.x86_64 : MPI implementation over Infiniband RDMA-enabled interconnect
rdma.noarch : Infiniband/iWARP Kernel Module Initializer
```

due to its peculiarity the infiniband network also requires the following package on at least one node that serves as a 'manager' for the latter ones.

```
opensm.x86_64 : OpenIB InfiniBand Subnet Manager and management utilities
```

The opensm daemon must be started on at least one node.

Though not required for operations one may find useful to install the following package for testing the infiniband connection:

```
ibutils.x86_64 : OpenIB Mellanox InfiniBand Diagnostic Tools
```

The ibutils package provides an instant check tool command that is

```
ibdiagnet
```

if everything is working properly it produces the following output:

```
-I- Discovering ... 4 nodes (1 Switches & 3 CA-s) discovered.
```

2.2.2 DiFX installation on vlbi-mgr

On the vlbi-mgr node the default installation directory was called

```
/soft
```

and contains

```
difx  fourfit_control  hops-3.8  ipp  machines  openmpi  pgplot
```

As seen in the contents of the /soft directory, we have installed the required IPP libraries in ipp/, the openmpi package in directory openmpi/ and also the pgplot package.

The required packages in Scientific Linux for the installation are:

```
bison.x86_64 : A GNU general-purpose parser generator
flex.x86_64 : A tool for creating scanners (text pattern recognizers)
fftw2.x86_64 : Fast Fourier Transform library
fftw2-devel.x86_64 : Headers, libraries and docs for the FFTW library
libtool.x86_64 : The GNU Portable Library Tool
expat.x86_64 : An XML parser library
expat-devel.x86_64 : Libraries and header files to develop applications using expat
automake16.noarch : A GNU tool for automatically creating Makefiles
autoconf.noarch : A GNU tool for automatically configuring source code
```

to install DiFX the first step taken is to edit the first lines of the setup.bash file found in the svn repository:

```
####### DIFX VERSION ########################
export DIFX_VERSION=DiFX-2.1.1

####### ROOT PATHS ##########################
export DIFXROOT=/soft/difx
export DIFX_PREFIX=$DIFXROOT
export PGPLOTDIR=/soft/pgplot
export IPPROOT=/soft/ipp

####### COMPILER ############################
export MPICXX=/soft/openmpi/bin/mpicxx
```

the next step is to source the modified setup.bash and run

```
./genipppc
```

if genipppc is able to find the IPP libraries the output will be the following:

```
Reading version from  /soft/ipp/ipp/include/ippversion.h
Got version  7.1.1
```

the final step to be taken is to launch

```
./install-difx
```

This command will execute all compilation tasks and install DiFX in the directory indicated in setup.bash as DIFXROOT.

Should HOPS package be included in the installation, then run

```
./install-difx --with-hops
```

although running this command leads to a more straightforward installation, this is not suggested because the latest HOPS package release is not always kept in line with DiFX releases. The HOPS inclusion also requires that pgplot has been correctly installed and its path is properly provided in setup.bash as PGPLOTDIR.

2.2.3 HOPS installation on vlbi-mgr

The following instructions are taken from the README file found inside the HOPS tarball [11].

If you haven't already done so, unpack the tarball in a convenient place:

```
    cd /Somewhere
    tar zxf hops-3.7.tar.gz
```

and proceed with the standard build process, e.g.:

```
    mkdir bld-3.7
    cd bld-3.7
    ../hops-3.7/configure
    make install
```

which installs the tools into a subdirectory of /Somewhere. You should use the –prefix=... option of configure if you want to install HOPS in some other directory (e.g. /usr/local). If the configure or install fails, do not despair: read on below in the next section of these instructions.

A bash setup script, hops.bash is created in the build directory (and copied to $HOME/bin if $HOME/bin exists). You can make an alias:

```
alias hops='source $HOME/bin/hops.bash'
```

and source it to set up your shell environment for HOPS work:

```
hops
```

The software is installed relative to a directory HOPS ROOT, which is synonymous with the install directory specified by the - -prefix option of configure. (It defaults to /Somewhere).

You can define HOPS ROOT in your environment or pass the assignment as an argument to configure if you wish to install the HOPS tools somewhere completely different. If you do this, then the /Somewhere directory (i.e. where you unpacked the tarball and built the tools) can be entirely deleted after 'make install'.

You may configure the software to build in the source directory (i.e. into the directory you untarred the software), but this is not recommended or required.

The complete software package requires X11, PGPLOT, (possibly) version 1.2 of the PNG library and ghostscript (compiled with X11 support). Unfortunately these are often not to be found in the usual places, so configure may need some help to get it right.

```
../hops-3.7/configure --help=short
```

provides the (short) help on configure. So if the default X11 search didn't work, one of these might find X11 on your system:

```
../hops-3.7/configure --enable-xtrap
../hops-3.7/configure --enable-pathx
```

Likewise, PGPLOT is normally installed in /usr/local/pgplot, but that may not be what you want:

```
../hops-3.7/configure PGPLOT_DIR=/usr/local/pgplot64
```

and if those fail, you can try setting more variables, e.g. the following works on MacOS with fink support for all the usual linux tools:

```
    ../hops-3.7/configure \
LDFLAGS=-L/usr/X11/lib \
CC=gcc-4 F77=gfortran \
X_INSANE='-lobjc -framework Foundation -L/sw/lib -lpng -laquaterm'
```

In particular the X INSANE variable is included last in the link step, (in fact, this gibberish works with the Fink MacOS pgplot package) so you can put all sorts of things here. The underlying issue is that there are too many ways that pgplot might have been compiled.

Note that if you set PGPLOT DIR as an environment variable, this directory should contain (at least) the three files:

```
libpgplot.a
libcpgplot.a
grfont.dat
```

and possibly pgxwin server if you're using X11 (which you almost certainly are unless you've got your own hacked-up version of PGPLOT).

Once you've successfully configured the package, 'make install' will build and install everything. If you want to break this into the two logical steps

```
make all
make install
```

2.3 Account setup

Once the /soft directory is ready to be exported a discretionary user has to be created on each node with the same user and group id, possibly the same used during the installation process. Our user has a bash profile setup that looks like this:

```
source /home/user/svn/DiFX-2.1/setup.bash
source hops-3.8/hops.bash
export CALC_SERVER=vlbi-mgr
export DEF_CONTROL=/soft/fourfit_control
export DIFX_MACHINES=/soft/machines
```

The full range of possible environment variables is available on a DiFX memo by Walter Brisken [1]. These include the case when no bash file is sourced so there are the whole lot of them directly specified.

3 Observational data (antenna files)

Observations are described and guided by the .vex (or .skd at its early stadium) produced by the PI with the help of the scheduling program. The .vex file is also the basis for the future scripts production during corraltion.

While observing each and every antenna will produce scan files (named also *Broadband data* in figures) that contain the astronomical observations data.

The scan files data format, which is described in a .vex section, depends on the particular backend the stations are equipped with.

Moreover every single antenna will give out a log file which describes the events that occurred during observation time.

The most challenging parts of achieving a correct correlation are obtaining the correct time delays from the antenna and checking you got the right association between the planned data format and the BBC frequencies assignments.

The following descriptions provide useful solutions to these problems but we do not guarantee that we have all of them covered. For instance, a new receiver at the antenna, or new hardware improvements, or peculiar cabling setups at the antenna could seriously mess with the common practices described so none of the following use cases should be taken for granted.

3.0.1 vex file

The vex file is the core component of an experiment. It comprises the most relevant informations both for the antenna operations and the correlator software. It comes out from sched [22] software for astronomical experiments and sked [6] for geodetic experiments, though sked standard output is a skd file (and older version of a vex file) and needs a conversion made by the same software for gaining a vex file. The antennas field system is capable of reading the older format though DiFX is not.

Usually the vex file is prepared by a PI who sets the experiment with either sched or sked and then generates the vex files to be sent at all the antennas involved and the correlator.

An astronomical vex file is made of different sections where the commented lines are preceded by *.

The EXPER section is a description of the experiment which contains a useful summary of what is going to be observed, the frequencies, the expected data rate and and time. These info rations come handy when setting up the v2d file.

```
$EXPER;
*
def apr15;
     exper_name = apr15;
     exper_description = "Test for Italian VLBI between Mc and Nt, IRA Difx";
     PI_name = "M. Giroletti";
     PI_email = giroletti@ira.inaf.it;
```

```
*    address:   INAF Istituto di Radioastronomia
*               Via Gobetti 101
*               40129 Bologna
*               Italy
*    phone:     +39 051 639 9394
*    during obs:+39 000 000000
*    fax:       +39 051 639 9431
*
*    year, doy: 2013, 105
*    date     : Mon 15 Apr 2013
*    MJD      : 56397
     exper_nominal_start=2013y105d12h00m00s;
     exper_nominal_stop=2013y105d16h00m00s;
*
     target_correlator = "OTHER IRADIFX";
*
*    integr_time    :     2.000 s
*    number_channels:    16
*    number_antenna :  2
*    cross_polarize : No
*    weight_func    : UNIFORM
*    distrib_medium : FTP
*    source_pos_cat : FROM SCHEDULE
*    distribute_to  :
*                     INAF Istituto di Radioastronomia
*                     Via Gobetti 101, 40129 Bologna
*                     Italy
*
enddef;
```

The MODE part that comes after the first section of the vex file summarizes the setups for the antenna. It contains the definitions for the sections that follow.

```
$MODE;
*
def sess113.C512;
     ref $PROCEDURES = Mode_01;
     ref $FREQ = 4966.49MHz16x8MHz:Mc;
     ref $FREQ = 4966.49MHz16x8MHz#02:Nt;
     ref $FREQ = 4966.49MHz16x8MHz#03:Nd;
     ref $IF = LO@4600MHzDPolTone/1:Mc;
     ref $IF = LO@4120MHzDPolTone/1:Nt;
     ref $IF = LO@4120MHzDPolTone/1#02:Nd;
     ref $BBC = 16BBCs:Mc;
     ref $BBC = 16BBCs#02:Nt;
     ref $BBC = 16BBCs#03:Nd;
     ref $TRACKS = MKIV.16Ch2bit1to2:Mc:Nt;
     ref $TRACKS = MARK5B.16Ch2bit1to1:Nd;
*    ref $HEAD_POS = DiskVoid <= obsolete definition
     ref $ROLL = NoRoll:Mc:Nt:Nd;
*    ref $PASS_ORDER = DiskVoid <= obsolete definition
     ref $PHASE_CAL_DETECT = DsbDetect:Mc:Nt:Nd;
enddef;
```

The following sections are about the geographical informations about the antennas involved in the observation and their tracking capabilities.

```
$SITE;
*
def MEDICINA;
     site_type = fixed;
     site_name = MEDICINA;
     site_ID = Mc;
*    elev=   67.16 long=-011:38:49. lat= 44:31:13.8
     site_position = 4461369.78560 m:  919597.03330 m: 4449559.32850 m;
     site_velocity = -0.018140   m/yr:  0.018770   m/yr:  0.011160  m/yr;
*    First line below is VEX standard format.  Use only when readers are ready.
*    site_position_epoch = 2008y001d;
     site_position_epoch =   54466;
enddef;
*
def NOTO;
     site_type = fixed;
     site_name = NOTO;
     site_ID = Nt;
*    elev=  143.22 long=-014:59:20. lat= 36:52:33.8
     site_position = 4934562.92650 m: 1321201.45870 m: 3806484.65980 m;
     site_velocity = -0.017650   m/yr:  0.017560   m/yr:  0.015030  m/yr;
*    First line below is VEX standard format.  Use only when readers are ready.
*    site_position_epoch = 2008y001d;
     site_position_epoch =   54466;
     horizon_map_az =  0.0 deg: 10.0: 30.0: 70.0:100.0:120.0:150.0:180.0:300.0:
                                310.0:360.0;
     horizon_map_el =  8.0 deg:  7.0:  6.0:  9.0:  6.5:  5.0:  6.0:  5.0:  5.0:
```

```
                                  6.5:  7.5;
enddef;
*
def NT_DBBC;
     site_type = fixed;
     site_name = NT_DBBC;
     site_ID = Nd;
*    elev=  143.22 long=-014:59:20. lat= 36:52:33.8
     site_position = 4934562.92650 m: 1321201.45870 m: 3806484.65980 m;
     site_velocity = -0.017650   m/yr:  0.017560   m/yr:  0.015030  m/yr;
*    First line below is VEX standard format.  Use only when readers are ready.
*    site_position_epoch = 2008y001d;
     site_position_epoch =   54466;
     horizon_map_az =  0.0 deg: 10.0: 30.0: 70.0:100.0:120.0:150.0:180.0:300.0:
                                310.0:360.0;
     horizon_map_el =  8.0 deg:  7.0:  6.0:  9.0:  6.5:  5.0:  6.0:  5.0:  5.0:
                                  6.5:  7.5;
enddef;
*------------------------------------------------------------------------------
$ANTENNA;
*
def MEDICINA;
     axis_type = az : el;
     antenna_motion = el :  30.0 deg/min :  3 sec;  *    0.310 deg/sec/sec
     antenna_motion = az :  45.0 deg/min :  3 sec;  *    0.820 deg/sec/sec
     axis_offset =    1.82700 m;
enddef;
*
def NOTO;
     axis_type = az : el;
     antenna_motion = el :  30.0 deg/min :  2 sec;  * 1000.000 deg/sec/sec
     antenna_motion = az :  43.0 deg/min :  2 sec;  * 1000.000 deg/sec/sec
     axis_offset =    1.83100 m;
enddef;
*
def NT_DBBC;
     axis_type = az : el;
     antenna_motion = el :  30.0 deg/min :  2 sec;  * 1000.000 deg/sec/sec
     antenna_motion = az :  43.0 deg/min :  2 sec;  * 1000.000 deg/sec/sec
     axis_offset =    1.83100 m;
enddef;
```

SOURCE section defines the sources observed during the experiment.

```
$SOURCE;
*
def 0235+164;
     source_name = 0235+164;
*    this source had calibrator code: V
*    alternate source name: J0238+1636
*    alternate source name: J0238+16
*    rfc_2011d Petrov, 2011, unpublished 64307 observations
     ra = 02h38m38.9301070s; dec =  16d36'59.274500"; ref_coord_frame = J2000;
*    ra = 02h35m52.6302148s; dec =  16d24'04.016081"; ref_coord_frame = B1950;
*    ra = 02h39m22.7835158s; dec =  16d40'18.131474"; ref_coord_frame = Date;
enddef;
*
def 3C84;
     source_name = 3C84;
*    this source had calibrator code: V
*    alternate source name: J0319+4130
*    alternate source name: 0316+413
*    alternate source name: J0319+41
*    GSFC 2011A astro solution     10290 Observations
     ra = 03h19m48.1600890s; dec =  41d30'42.104090"; ref_coord_frame = J2000;
*    ra = 03h16m29.5672587s; dec =  41d19'51.916964"; ref_coord_frame = B1950;
*    ra = 03h20m40.6987116s; dec =  41d33'31.166613"; ref_coord_frame = Date;
enddef;
*
def BLLAC;
     source_name = BLLAC;
*    this source had calibrator code: V
*    alternate source name: J2202+4216
*    alternate source name: 2200+420
*    alternate source name: J2202+42
*    alternate source name: VR422201
*    rfc_2011d Petrov, 2011, unpublished 55580 observations
     ra = 22h02m43.2913710s; dec =  42d16'39.979870"; ref_coord_frame = J2000;
*    ra = 22h00m39.3625043s; dec =  42d02'08.590734"; ref_coord_frame = B1950;
*    ra = 22h03m16.1962786s; dec =  42d20'23.299426"; ref_coord_frame = Date;
enddef;
*
def J2347+5142;
     source_name = J2347+5142;
```

```
*    this source had calibrator code: V
*    alternate source name: 2344+514
*    GSFC 2011A astro solution        36 Observations
     ra = 23h47m04.8367000s; dec =  51d42'17.881650"; ref_coord_frame = J2000;
*    ra = 23h44m36.2593684s; dec =  51d25'37.902128"; ref_coord_frame = B1950;
*    ra = 23h47m43.8885784s; dec =  51d46'36.903954"; ref_coord_frame = Date;
enddef;
```

The SCHED section of the vex file is relative to the scheduled scans that will be observed. The sections about frequencies and the bitstreams or 'TRACKS' that define the data formats will be discussed later.

```
$SCHED;
* schedule section for experiment apr15
* Test for Italian VLBI between Mc and Nt, IRA Difx
scan No0001;
*    start=2013y105d12h00m00s <= original start, modified for tape start.
     start=2013y105d11h59m55s; mode=sess113.C512; source=J2347+5142;
*              :data_good:data_stop:goto_foot: pass:  wrap :driv:tape at
     station=Mc:    5 sec:  245 sec:    0.000 GB:   :       : 1;
     station=Nt:    5 sec:  245 sec:    0.000 GB:   :       : 1;
     station=Nd:    5 sec:  245 sec:    0.000 GB:   :       : 1;
endscan;
scan No0002;
*    start=2013y105d12h04m30s <= original start, modified for tape start.
     start=2013y105d12h04m25s; mode=sess113.C512; source=J2347+5142;
     station=Mc:    0 sec:  245 sec:   15.805 GB:   :       : 1;
     station=Nt:    0 sec:  245 sec:   15.805 GB:   :       : 1;
     station=Nd:    0 sec:  245 sec:   15.805 GB:   :       : 1;
endscan;
scan No0003;
*    start=2013y105d12h09m14s <= original start, modified for tape start.
     start=2013y105d12h09m09s; mode=sess113.C512; source=BLLAC;
     station=Mc:    5 sec:  245 sec:   31.611 GB:   :       : 1;
     station=Nt:    1 sec:  245 sec:   31.611 GB:   :       : 1;
     station=Nd:    1 sec:  245 sec:   31.611 GB:   :       : 1;
endscan;

...
```

3.0.2 antenna log files

The antenna log files come out of the antennas Mark IV Field System [13]. The vex file provided can automatically govern the experiment planned. While operating the most important outputs for correlation from the antennas are the tsys and the gps-fmout reports.

A log file also gives out relevant feedback about the equipment used and correct BBC assignments. It may as well tell whether a source was not correctly observed and why.

```
2013.158.05:46:37.54;Log Opened: Mark IV Field System Version 9.10.4
2013.158.05:46:37.54;location,MEDICINA,-11.65,44.52,12.5
2013.158.05:46:37.54:"      MEDICINA Mark5A
2013.158.05:46:37.54:" drudg version 2008Oct08 compiled under FS  9.10.04
2013.158.05:46:37.54:" Rack=Mark4     Recorder 1=Mark5A    Recorder 2=Mark5A
```

System temperature, the noise in the system, is a combination of noise from various sources:

$$T_{sys} = T_{receiver} + T_{ground} + T_{sky}$$

System temperatures can vary unpredictably during a VLBI experiment due to changes in the receiver temperature, the spill-over, RFI etc. and so must be monitored continuously

A secondary calibration source (usually a broad-band noise cal signal) of constant noise temperature T_{cal} is periodically injected and the change in total power is compared to the power measured when this cal signal is switched off. From these measurements the system temperature Tsys can be derived via:

$$T_{sys} = \frac{T_{cal} P_{cal-off}}{P_{cal-on} - P_{cal-off}}$$

$P_{cal-off}$ is when there is no phase calibration signal injected, whereas P_{cal-on} implies the presence of the signal.

The following is an example of a log line retrieving the T_{sys} temperature in Kelvin that could be useful to double-check with the antenna operator if everything worked properly.

```
2013.158.06:00:07.87/tsys/1d,72.9,3d,80.0,5d,79.3,7d,85.2,i1,77.7
```

The antenna clock can be found in the log file, but also on the IRA ftp-hosted files [10] that are updated daily. The time reference, usually displayed in microseconds, has got a sign that is minus but it has to be converted to plus when inserted into the v2d file. It is to be noted that there is no clock reference in the logs produced by dBBCs.

Even though the sign problem appears to be a minor issue, it is of the highest importance when trying to find fringes. In fact the delay window used to find fringes it is sensible to the sign, the risk is not to choose the appropriate one and miss the fringe.

```
2013.158.06:00:09.08/gps-fmout/2.9230e-06
```

The trackform parts provide useful informations about the bitstream mapping in the scan file, though this is not found in the log while operating with a dBBC. The l stands for lower, u for upper and s for sign, m for magnitude.

```
2013.158.05:46:40.57&trkf01/trackform=2,2ls,6,2lm,10,1ls,14,1lm,18,2us,22,2um
2013.158.05:46:40.57&trkf01/trackform=3,4ls,7,4lm,11,3ls,15,3lm,19,4us,23,4um
2013.158.05:46:40.57&trkf01/trackform=102,6ls,106,6lm,110,5ls,114,5lm,118,6us
2013.158.05:46:40.57&trkf01/trackform=130,5um,103,8ls,107,8lm,111,7ls,115,7lm
2013.158.05:46:40.57&trkf01/trackform=127,7us,131,7um
```

The trackform lines are related to the vex section about TRACKS and their assignment as bitstreams on scan files, so the whole subject further explained in the Data formats section.

3.0.3 Scan files

In this section it is described the data that gets delivered to the correlator. The data is organized in UNIX files usually the common size between 15 up to 50 GB. The data streams can be recorded at different speeds, the most common being 1 Gbit/sec. The scan files are usually the portion of an experiment when the antenna is stably tracking a source and recording the signals. They are indicated in the vex file as

```
scan No0002;
*    start=2013y105d12h04m30s <= original start, modified for tape start.
     start=2013y105d12h04m25s; mode=sess113.C512; source=J2347+5142;
     station=Mc:    0 sec:  245 sec:   15.805 GB:   :       : 1;
     station=Nt:    0 sec:  245 sec:   15.805 GB:   :       : 1;
     station=Nd:    0 sec:  245 sec:   15.805 GB:   :       : 1;
endscan;
```

The recording format may vary depending on whether the vlbi-terminal is MARK IV, MARK5B or a dBBC. The actual de-facto standard is MARK5B format which guarantees the best interoperability between systems.

The scan files recorded onto a MARK5 system reside in a proprietary filesystem, and to make them readable on a Linux OS they need to be extrapolated from that. The usual file name contains the experiment name followed by a two letter antenna code and a progressive number of the scan.

```
<experiment_name>_<antenna_code>_<scan_number>
```

There are tools in DiFX to check if the scan was recorded properly, one is *directory2filelist* or *m5d*, the first one tries to read the scans and produces an output list of the files to be included in the v2d file, the latter can decode Mark3/4 and Mark5B formats.

There is also a basic check of the file that can be run using od.

```
od -tx4 n1312_mc_no0011

0000000 abaddeed bead0000 45024601 000060bb
0000020 94a086a2 999639f6 a8a1ef59 f39dcafb
0000040 568563bc ac7ea936 e96d82c6 becba216
0000060 76369097 28b67935 6680c68a b44ac1c4
0000100 309586a2 6659a75e 484f985a f5c5ae4f
0000120 3b15a94a 4791add0 6ca34aa4 97e35c9d
....
```

3.1 Data formats

Recently a breaking point has come with the introduction of a new kind of VLBI terminal, the dBBC [7]. The main idea underlying the 'dBBC' project is to replace the existing VLBI terminal with a complete and compact system to be used with any VSI compliant recorder or data transport, challenging the now in use MARK5s and MARKIVs.

3.1.1 "MARKs" data

The Mark5 development program, which started with the Mark 5A at 1 Gbps, has now evolved to next-generation Mark 5A+/Mark 5B/Mark 5B+ systems, and was expanded in 2007 to include the Mark 5C. Each generation has expanded and extended the capabilities, extending to 4 Gbps with the Mark 5C, and all generations are compatible with standard Mark 5 disk modules. All of the Mark 5 systems have been jointly developed with Conduant Corporation of Longmont, which provides the underlying real-time disk-recording technology. Conduant also makes Mark 5 systems available to the VLBI community at significant discounts from standard commercial prices. A brief summary of each of the Mark 5 generations:

Mark5A: The Mark 5A system, released in 2003, is intended as a direct replacement for a Mark 4 or VLBA magnetic-tape transport at either a station or a correlator. It records 8, 16, 32 or 64 tracks from a Mark4/VLBA formatter, and plays back in the same Mark4/VLBA format. As such, the Mark 5A is a direct replacement for a Mark4 tape unit at 1024 Mbps and VLBA tape unit at 512 Mbps. The disks are organized into two '8-pack' modules which may be used in a ping-pong fashion for near-continuous recording and playback.

Mark5B: The Mark 5B is VSI-H-compliant system with capability up to 1024 Mbps; no external formatter is necessary. The system will also support several backwards-compatibility modes with existing Mark4/VLBA correlator systems. The Mark 5B has been in service since the summer 2006.

Mark 5A+: The Mark 5A+ is identical to Mark 5A in hardware, but has had its FPGA code augmented to support playback of disk modules recorded on Mark 5B. In this special backwards-compatible playback mode, the Mark 5B data are transformed into VLBA tape-format data streams which can be directly accepted by Mark 4 correlators. The Mark 5A+ became operational in 2006 and is used at some Mark 4 correlators.

Mark5B+: The Mark 5B+ is a VSI-H-compliant system with recording capability extended to 2048 Mbps. The Mark 5B+ is in all other respects identical in capability to the Mark 5B. The Mark 5B+ has been in service since late 2006.

Mark5C: The Mark5C is a new system under development with the capability to support up to 4096Mbps from a 10 Gigabit Ethernet data source; in parallel developments, new Digital Backend systems are being developed as data sources for the Mark 5C in the U.S. and Europe. Playback, as well, is via a 10 Gigabit Ethernet data stream, which is compatible with ongoing software-correlator developments in Australia, Europe and the U.S. The Mark 5C is fundamentally a formatless packet recorder, though a parallel Mark 5C data-format specification defines a suggested standard VLBI data format. In a departure from earlier Mark 5 generations, as well as almost all previous VLBI systems, the Mark 5C breaks the normal constraint of 2n frequency channels, allowing more flexibility in recording and playback. In addition, for backwards compatibility, the Mark 5C can be configured to write data to disk in a Mark 5B-compatible data format that allows playback on standard Mark 5B/5B+ systems.

The "Marks" date back to a time when tapes were in use, so there were different track numbers (or stream of bits) assigned depending on the recording media. Nowadays disks are in production everywhere and the scans are stored onto files that have a word of 32 bits. Due to the two bits reserved by the tape recording systems at the beginning, the numbering of bits or so-called 'Tracks' start at two and end at 33.

To get an insight view of how is the scan file written, we need to get back to the vex file where the BBCs are mapped to reconstruct the right track assignment

```
$FREQ;
*
def 4966.49MHz16x8MHz;
* mode =  1    stations =Mc
     sample_rate =  16.000 Ms/sec;  * (2bits/sample)
     chan_def = :  4966.49 MHz : L :   8.00 MHz : &CH01 : &BBC02 : &L_Cal; *Rcp
     chan_def = :  4966.49 MHz : L :   8.00 MHz : &CH02 : &BBC01 : &L_Cal; *Lcp
     chan_def = :  4966.49 MHz : U :   8.00 MHz : &CH03 : &BBC02 : &L_Cal; *Rcp
     chan_def = :  4966.49 MHz : U :   8.00 MHz : &CH04 : &BBC01 : &L_Cal; *Lcp
     chan_def = :  4982.49 MHz : L :   8.00 MHz : &CH05 : &BBC04 : &L_Cal; *Rcp
     chan_def = :  4982.49 MHz : L :   8.00 MHz : &CH06 : &BBC03 : &L_Cal; *Lcp
     chan_def = :  4982.49 MHz : U :   8.00 MHz : &CH07 : &BBC04 : &L_Cal; *Rcp
     chan_def = :  4982.49 MHz : U :   8.00 MHz : &CH08 : &BBC03 : &L_Cal; *Lcp
     chan_def = :  4998.49 MHz : L :   8.00 MHz : &CH09 : &BBC06 : &L_Cal; *Rcp
     chan_def = :  4998.49 MHz : L :   8.00 MHz : &CH10 : &BBC05 : &L_Cal; *Lcp
     chan_def = :  4998.49 MHz : U :   8.00 MHz : &CH11 : &BBC06 : &L_Cal; *Rcp
     chan_def = :  4998.49 MHz : U :   8.00 MHz : &CH12 : &BBC05 : &L_Cal; *Lcp
     chan_def = :  5014.49 MHz : L :   8.00 MHz : &CH13 : &BBC08 : &L_Cal; *Rcp
     chan_def = :  5014.49 MHz : L :   8.00 MHz : &CH14 : &BBC07 : &L_Cal; *Lcp
     chan_def = :  5014.49 MHz : U :   8.00 MHz : &CH15 : &BBC08 : &L_Cal; *Rcp
```

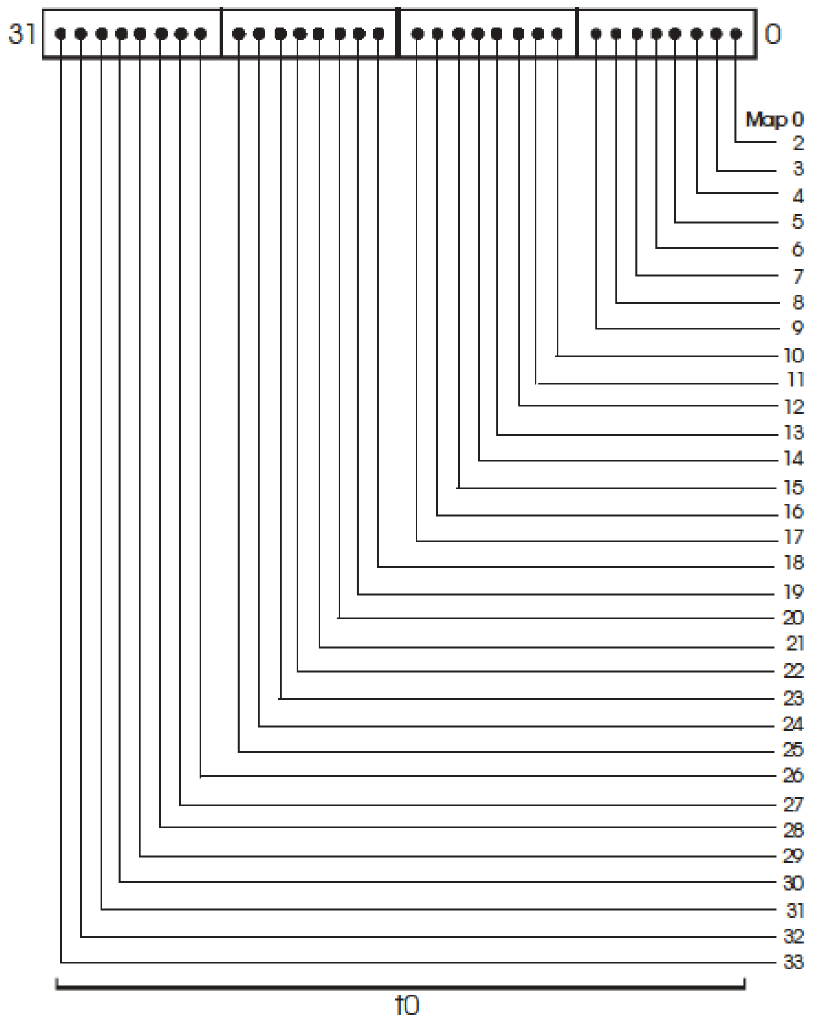


Figure 3: Mapping of bits in FPDP word to tracks for all cases of 32 output tracks. t_0 *refers to a single output-track clock cycle. [23]*

```
     chan_def = :  5014.49 MHz : U :   8.00 MHz : &CH16 : &BBC07 : &L_Cal; *Lcp
enddef;
```

Here we can see that there are 4 frequencies scanned: 4966.49 MHz, 4982.49 MHz, 4998.49 MHz and 5014.49 MHz. Each of those is divided into 2 sub-groups for the MARKIV video-converter in lower side bands and upper side bands where a bit indicates the sign and the other bit indicates the magnitude ± 220 mv. It is important also to note the L and U letters that stand for upper or lower, each of these frequencies will gain two representations when it comes to the subsequent track assignment for the sign and magnitude.

```
$TRACK
*
def MKIV.16Ch2bit1to2;
* mode =  1    stations =Mc:Nt
*    format = MKIV1:2, and fan-out = 2
*    mode requires   8.00Mb/s/tr; stations using disks
     track_frame_format = Mark4;
     data_modulation = off;
     fanout_def =   : &CH01 : sign : 1:  2:  4;
     fanout_def =   : &CH01 :  mag : 1:  6:  8;
     fanout_def =   : &CH02 : sign : 1: 10: 12;
     fanout_def =   : &CH02 :  mag : 1: 14: 16;
     fanout_def =   : &CH03 : sign : 1: 18: 20;
     fanout_def =   : &CH03 :  mag : 1: 22: 24;
     fanout_def =   : &CH04 : sign : 1: 26: 28;
     fanout_def =   : &CH04 :  mag : 1: 30: 32;
     fanout_def =   : &CH05 : sign : 1:  3:  5;
     fanout_def =   : &CH05 :  mag : 1:  7:  9;
     fanout_def =   : &CH06 : sign : 1: 11: 13;
     fanout_def =   : &CH06 :  mag : 1: 15: 17;
     fanout_def =   : &CH07 : sign : 1: 19: 21;
     fanout_def =   : &CH07 :  mag : 1: 23: 25;
     fanout_def =   : &CH08 : sign : 1: 27: 29;
     fanout_def =   : &CH08 :  mag : 1: 31: 33;
     fanout_def =   : &CH09 : sign : 2:  2:  4;
     fanout_def =   : &CH09 :  mag : 2:  6:  8;
     fanout_def =   : &CH10 : sign : 2: 10: 12;
     fanout_def =   : &CH10 :  mag : 2: 14: 16;
     fanout_def =   : &CH11 : sign : 2: 18: 20;
     fanout_def =   : &CH11 :  mag : 2: 22: 24;
     fanout_def =   : &CH12 : sign : 2: 26: 28;
     fanout_def =   : &CH12 :  mag : 2: 30: 32;
     fanout_def =   : &CH13 : sign : 2:  3:  5;
     fanout_def =   : &CH13 :  mag : 2:  7:  9;
     fanout_def =   : &CH14 : sign : 2: 11: 13;
     fanout_def =   : &CH14 :  mag : 2: 15: 17;
     fanout_def =   : &CH15 : sign : 2: 19: 21;
     fanout_def =   : &CH15 :  mag : 2: 23: 25;
     fanout_def =   : &CH16 : sign : 2: 27: 29;
     fanout_def =   : &CH16 :  mag : 2: 31: 33;
enddef;
```

This kind of assignment can be also retrieved from the antenna log file in the *trackform* lines where a MARKIV formatter is in production. This does not apply when a dBBC is in use. When in doubt one should check if the *trackform* lines are coherent with the $TRACK and $FREQ sections of the vex file.

```
trackform=2,2ls,6,2lm,10,1ls,14,1lm,18,2us,22,2um,26,1us,30,1um
trackform=3,4ls,7,4lm,11,3ls,15,3lm,19,4us,23,4um,27,3us,31,3um
trackform=102,6ls,106,6lm,110,5ls,114,5lm,118,6us,122,6um,126,5us
trackform=130,5um,103,8ls,107,8lm,111,7ls,115,7lm,119,8us,123,8um
trackform=127,7us,131,7um
```

Table 1 shows how to work out the Mark5B bitstreams assignment to decode them when correlating as established by Haystack. Differences can be found when correlating geodetic observations as mainly the upper side bands are kept into considerations and only a single channel lower magnitude is taken into consideration for verification purposes.

Mark 5B bit-stream	**Astronomy mode**	**Geodesy mode**	**VSI2**
0	1US	1US	9US
1	1UM	1UM	9UM
2	2US	2US	10US
3	2UM	2UM	10UM
4	3US	3US	11US
5	3UM	3UM	11UM
6	4US	4UM	12US
7	4UM	4UM	12UM
8	5US	5US	13US
9	5UM	5UM	13UM
10	6US	6US	14US
11	6UM	6UM	14UM
12	7US	7US	-
13	7UM	7UM	-
14	8US	8US	-
15	8UM	8UM	-
16	1LS	1LM	9LS
17	1LM	1LM	9LM
18	2LS	8LS	10LS
19	2LM	8LM	10LM
20	3LS	9US	11LS
21	3LM	9UM	11LM
22	4LS	10US	12LS
23	4LM	10UM	12LM
24	5LS	11US	13LS
25	5LM	11UM	13LM
26	6LS	12US	14LS
27	6LM	12UM	14LM
28	7LS	13US	-
29	7LM	13UM	-
30	8LS	14US	-
31	8LM	14UM	-

Table 1: Mark5B bitstream assignment [23]

3.1.2 dBBC data

The main idea laying behind to the 'DBBC' project is to replace the existing VLBI terminal with a complete and compact system to be used with any VSI compliant recorder or data transport.

The entire project is based on a flexible architecture, composed by one or more FPGA boards as computation elements, placed in a mixed cascaded/parallel structure, so to guarantee a parallel usage of data input and a shared parallel output data flow. In a DBBC, a single system unit is composed by four RF/IF Input in the ranges 0.01-512, 512-1024, 1024-1536, 1536-2048 MHz, 2048-2100 MHz with each of them feeding a 1.024 GHz clock sampler. Then four polarizations or bands are available for a single group of output channels selection. A group of 64 channels is able to handle a shared combination of channels coming from the four bands, supporting two VSI output connectors as output [7].

The Fila10G board can be installed either internally or externally in a dBBC, it provides the output data needed for correlation.

It has two optical outputs that can be used for network streaming or through a glass-to-copper device (GLAPPER) to save the data onto a MARK5 system. This means technically that the device converts the data stream from an XFP transceiver to a CX4 connector.

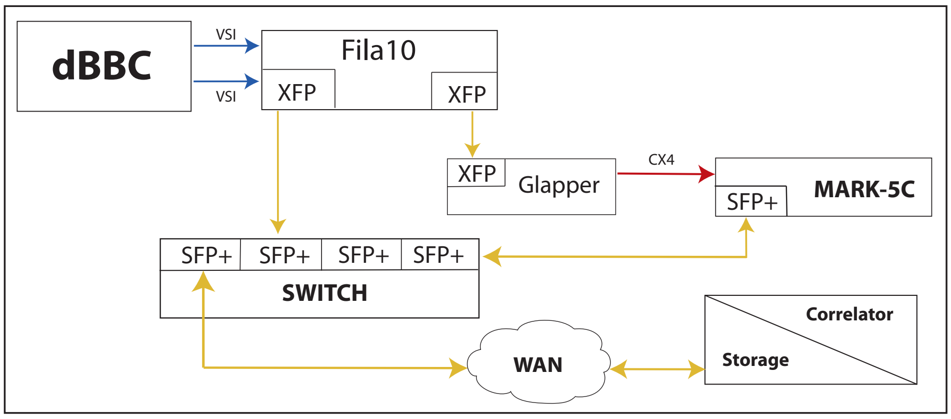


Figure 4: Network connection schema of a dBBC as designed in Medicina station

3.1.3 Network data formats

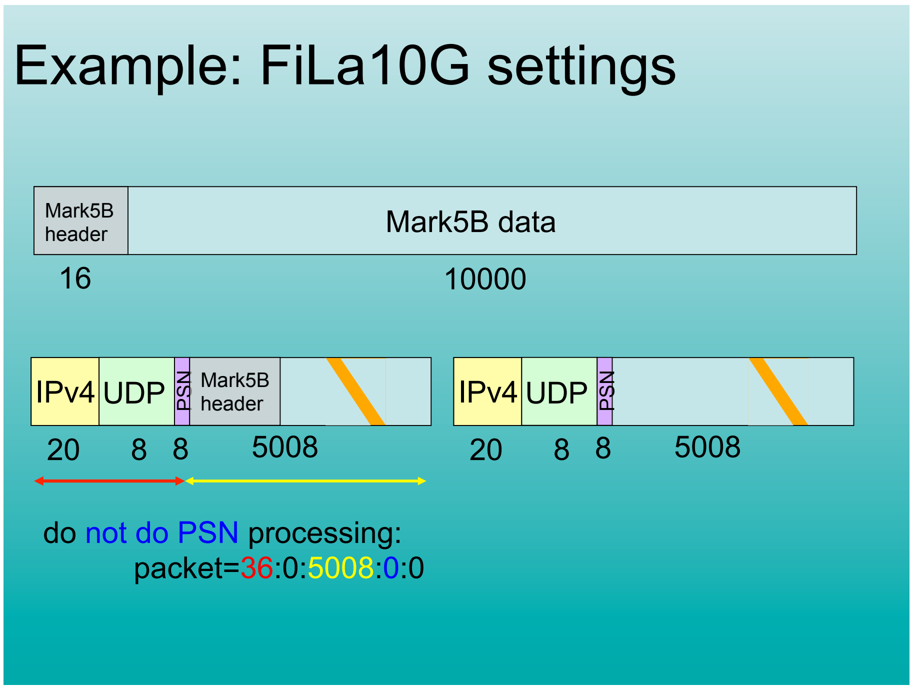
Figure 5 remarks clearly what the main issue is when trying to stream MARK5B formatted data onto the network. The word size is of 10016 so even with jumbo frames (9000) it would be impossible to achieve this task. The decision was taken to split into two parts the word packet to come up with two 5016 packets each containing a PSN (Packet Serial Number) that allows a correct ordering. In case one of the two packets is lost, the values for that word should be automatically zeroed to assure a correct ordering of the previous and following packets.

To avoid this issue related to the MARK5B format, a new data kind is bound to be introduced in the near future, that is VDIF. VLBI Data Interchange Format should overcome this, as the user will be able to set the packet size on a FILA10G according to the link traffic and speed, enabling also more flexibility to organize real-time correlations with DiFX.

The VDIF specification is based upon a basic self-identifying Data Frame, which carries a time segment of time-sampled data from one or more frequency sub-bands. The length of a Data Frame may be chosen by the user to best match the chosen transport protocol; for example, in the case of real-time network transfer, a VDIF Data Frame length would normally be chosen so that exactly one Data Frame is carried by each on-the-wire packet. It is important to emphasize that the VDIF Data Frame is fundamentally transport-protocol independent, so that exactly the same set of Data Frames can represent VLBI data through a network transfer or be stored to a physical disk file.

Unfortunately this format is still under intensive development at the moment. An agreement has to be reached about its endianess, because some corraltors still prefer a big-endian format, whereas DiFX was recently developed onto more nowadays common little-endian architectures.

When a dBBC is in use there are significant alterations to the track assignment caused by the different internal arrangements of the signal channels on the FPGA boards. Although the assignment is still following the MARK5B scheme shown in Table 1 the bit streams do get mixed up and the beginning of their count now starts at two, still beholding the reserved first two bits for the legacy tape recording system.



do not do PSN processing:
packet=36:0:5008:0:0

Figure 5: Fila10G Mark5B net packets compared to default ones [21]

bit-stream (2-33)	**Astronomy mode**
18	1LS
19	1LM
26	5LS
27	5LM
2	1US
3	1UM
10	5US
11	5UM
20	2LS
21	2LM
28	6LS
29	6LM
4	2US
5	2UM
12	6US
13	6UM
22	3LS
23	3LM
30	7LS
31	7LM
6	3US
7	3UM
14	7US
15	7UM
24	4LS
25	4LM
32	8LS
33	8LM
8	4US
9	4UM
16	8US
17	8UM

Table 2: dBBC bitstream assignment

4 Correlation pipeline

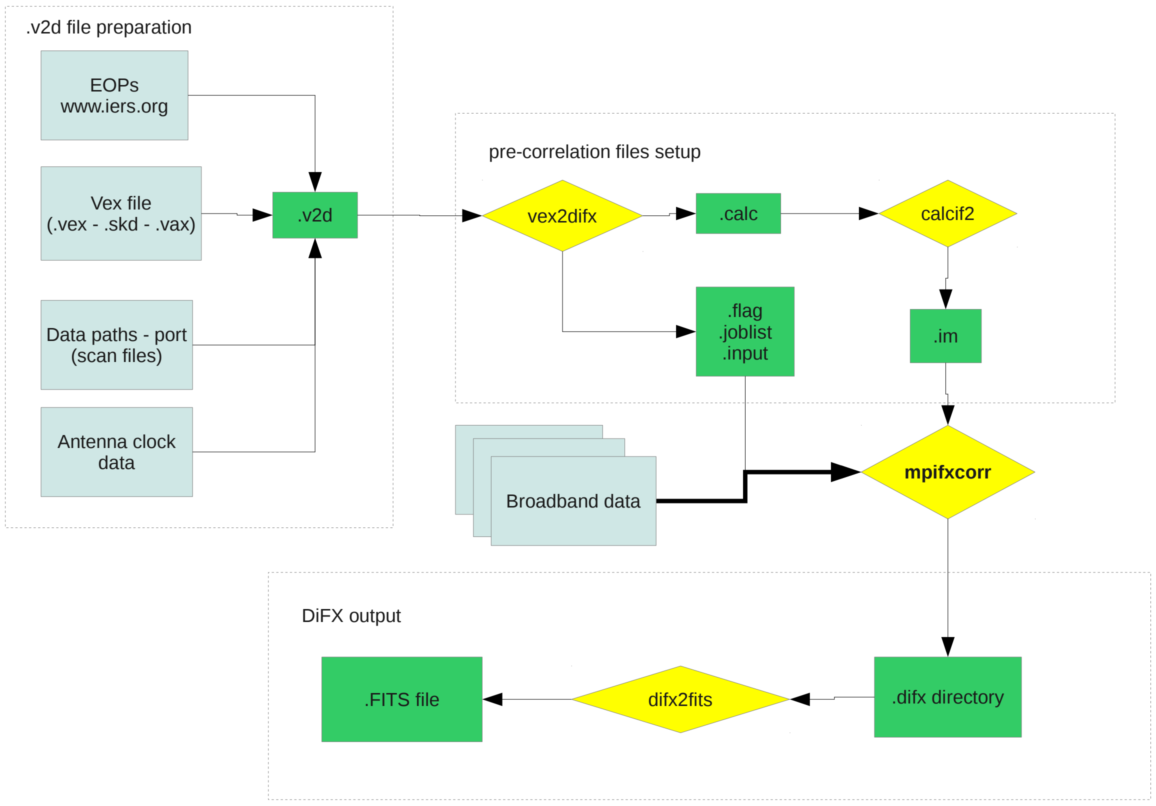


Figure 6: DiFX correlation pipeline: the grey parts represent the input, the green ones the output produced after each phase

In this section it is described the full correlation pipeline to be set for a vlbi experiment. In-depth details are provided for the data formats currently in use and the best practices to get a correct output.

4.1 Correlation core

After discussing all the relevant informations needed to achieve a correlation task, we are now going to get into practical details. The following paragraphs will discuss all the relevant parts of a correlation, beginning from writing a correct v2d file, moving to the preprocessing tasks and finally getting to the mpifxcorr execution which would eventually produce a usable output for astronomers or geodesists alike.

4.1.1 EOP

EOPs are made up of several components.

Universal time (UT1) stands for the earth rotation, which performs one revolution in about 24h. The earth rotation is uneven, so UT is not linear with respect to atomic time. It is practically proportional to the sidereal time, which is also a direct measure of earth rotation.

Due to the very slow pole motion of the earth, the Celestial Ephemeris Pole (CEP, or celestial pole) does not stick still on surface of earth. The Celestial Ephemeris Pole is *calc* observation data, and is somehow averaged, so it differs from the instantaneous rotation axis by quasi-diurnal terms, which are as small as under 0.01". In setting up a coordinate system, a static terrestrial point called the IERS Reference Pole, or IRP, is used as origin; the x-axis is in the direction of IRM, the IERS Reference Meridian; the y-axis is in the direction 90 degrees West longitude. x and y are the coordinates of the CEP relative to the IRP

Celestial pole offsets are described in the IAU Precession and Nutation models. The observed differences with respect to the conventional celestial pole position defined by the models are monitored and reported by the IERS [8].

Fortunately there is a quick way to retrieve all the the relevant data for a correlation. In the espresso package provided in DiFX under applications/espresso there is a python tool called getEOP.py which connects to the IERS database and once run given the MJD date of the experiment, which is found in the vex file, this is the following output:

```
./getEOP.py 56453

Fetching EOP data...
...got it.
Fetching Leap second data...
...got it.

#EOPs downloaded at 2013-06-20 17:57:16 (UTC)
EOP 56451 { xPole=0.115640 yPole=0.402970 tai_utc=35 ut1_utc=0.070633 }
EOP 56452 { xPole=0.116380 yPole=0.402120 tai_utc=35 ut1_utc=0.070534 }
EOP 56453 { xPole=0.117310 yPole=0.401450 tai_utc=35 ut1_utc=0.070432 }
EOP 56454 { xPole=0.118260 yPole=0.401060 tai_utc=35 ut1_utc=0.070258 }
EOP 56455 { xPole=0.119280 yPole=0.400640 tai_utc=35 ut1_utc=0.070006 }
Processed 12565 lines
```

These lines can be directly inserted in the v2d file that summons the pre-correlation data. The EOPs fetched are two days before and after the given MJD experiment day.

4.1.2 v2d file

The v2d file represents the entry point when starting a correlation. We have produced an example to make it more understandable. The full reference of options and tweaks is available online [5].

```
vex = vlbit.vex
```

```
tweakIntTime = True
ANTENNA NT {
        clockOffset=1.68
        clockRate=-1.060e-12
        clockEpoch=2013y105d14h30m00s
        filelist=/space2/vlbit/full_scans/ntlist
}
ANTENNA ND {
        clockOffset=0
        clockRate=-1.060e-12
        clockEpoch=2013y105d14h30m00s
        filelist=/space2/vlbit/full_scans/ndlist
}
ANTENNA MC {
        clockOffset=12.30
        clockRate=1.687e-12
        clockEpoch=2013y105d14h30m00s
        filelist=/space1/vlbit/apr15_mc/mclist
}
SETUP ita
{
  tInt =  2.000
  nChan = 128
  doPolar = False # Full stokes
}
RULE clox
{
  scan = No0010
  setup = ita
}

EOP 56395 { xPole=0.055420 yPole=0.385060 tai_utc=35 ut1_utc=0.138215 }
EOP 56396 { xPole=0.055440 yPole=0.385670 tai_utc=35 ut1_utc=0.137033 }
EOP 56397 { xPole=0.055570 yPole=0.386040 tai_utc=35 ut1_utc=0.135929 }
EOP 56398 { xPole=0.055880 yPole=0.386440 tai_utc=35 ut1_utc=0.134822 }
EOP 56399 { xPole=0.056490 yPole=0.386940 tai_utc=35 ut1_utc=0.133680 }
```

What the v2d file is specifying to the DiFX correlator is that it has to take the necessary informations from a vex file, that we have the data from three antennas available, we want to integrate every 2 seconds using 128 channels and we only need a single scan while doing this. Moreover we provide the EOP informations relative to the MJD date of 56397 that is the day the observation was run.

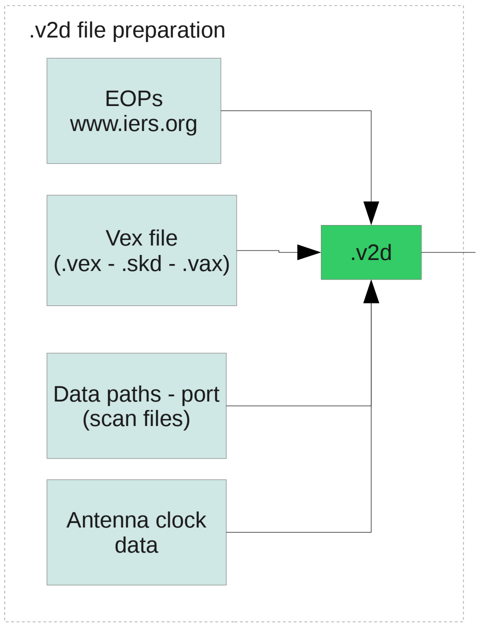


Figure 7: Build up the v2d file

The tool that helps us compose the v2d file is *getEOPs*. Once given the MJD date of the experiment it fetches the two previous and upcoming days EOPs from IERS database. They are subject to change if fetched later as they undergo constant revisions and future dates are only estimates.

To populate a full list of the scan files relative to an observation that are stored under a certain directory one can simply run *directory2filelist*.

```
directory2filelist /space1/Fila10Mark5Bnet/ Mark5B-1024-16-2
```

Unless the data format specified in the last option is not recognized, the screen output should contain all the full paths of the scan present in the specified directory passed as the first option. These can be piped into a file that will eventually be specified as filelist= into the v2d ANTENNA sections.

4.1.3 Antenna clock data

One of the most delicate part when it comes to correlate astronomical data is finding the right clock informations about each station. There is a tool that extracts all GPS measurements from the antennas log files and makes a least- squares fit to the readings to determine a delay and rate for the clock model. Two conventions are possible for the sign of the clock as gps-fmout or fmout-gps. There is also another possible trap that one might fall into that is sometimes microseconds are recorded, sometimes seconds in scientific notation. The *log2clock* program should take care of all these problems but the sign convention may not be guaranteed to be right.

This is why it is sensible to double check on the database held at IRA [10] where the clock offsets and rates are regularly reported everyday for most of the EVN stations. The sign reported there HAS TO be inverted when written into the v2d file.

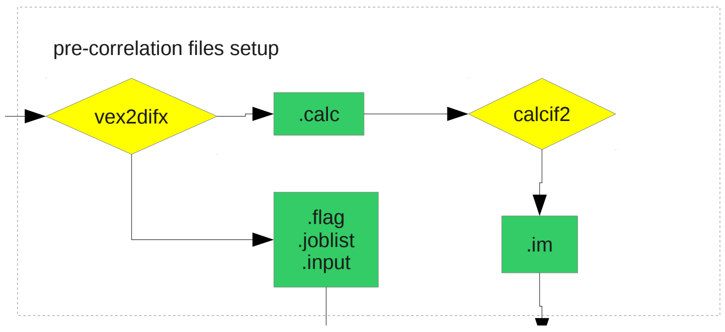


Figure 8: Generating the pre-correlation files

4.1.4 vex2difx processing

Finally the v2d file is ready to be processed by *vex2difx*. If no mistakes were made it has to produce 4 output files.

```
vex2difx vlbit.v2d
```

```
vlbit_1.calc
vlbit_1.flag
vlbit.joblist
vlbit_1.input
```

The .flag file contains lists of antenna-based flags generated by the on-line system that should be applied to the visibility data. This file contains two kinds of lines. Flag lines always consist of exactly 5 fields:
1. antId : Station name abbreviation, e.g., Mc
2. start : Beginning of flagged period (day of year, including fractional portion)
3. end : End of flagged period (day of year, including fractional portion)
4. recChan : Record channel affected; -1 for all record channels, otherwise a zero-based index
5. reason : Reason for flag, enclosed in single quotes, truncated to 24 characters
The flag rows are sorted first by antenna, and then start time [2].

A single .joblist file is written by *vex2difx* as it produces the DiFX .input (and other) files for a given correlator pass. This file contains the list of jobs to run and some versioning information that allows improved accountability of the software versions being used. This file is used by difxqueue and makefits to ensure that a complete set of jobs is accounted for. The file is composed of two parts: a header line and one line for each job. The header line consists of

a series of key=value pairs. Each key and value must have no whitespace and no whitespace should separate these words from their connecting = sign. While any number of key-value pairs may be specified, the following ones (which are case sensitive) are expected to be present:
1. exper : the name of the experiment, including the segment code
2. v2d : the vex2difx input file used to produce the jobs of this pass
3. pass : the name of the correlator pass
4. mjd : the modified Julian day when vex2difx created this file
5. DiFX : the version name for the DiFX deployment (the value of $DIFX VERSION when vex2difx was run)
6. vex2difx : the version of vex2difx that was run

Each additional line contains information for one job in the pass. The columns are:
1. jobName : the name/prefix of the job
2. mjdStart : the observe start time of the job
3. mjdStop : the observe stop time of the job
4. nAnt : the number of antennas in the job
5. tOps : estimated number of trillion floating point operations required to run the job
6. outSize : estimated FITS file output size (MB).

Finally the .input file is passed to *mpifxcorr*. Before that it is necessary to calculate the geometric delays and atmospheric delays required by DiFX with *calcif2*, but .input contains the most relevant informations to be used for correlating the data. The following is an extract of the content.

```
# COMMON SETTINGS ##!
CALC FILENAME:      /space2/fullVlbit/vlbit_1.calc
CORE CONF FILENAME: /space2/fullVlbit/vlbit_1.threads
EXECUTE TIME (SEC): 245
START MJD:          56397
START SECONDS:      46105
ACTIVE DATASTREAMS: 3
ACTIVE BASELINES:   3
VIS BUFFER LENGTH:  80
OUTPUT FORMAT:      SWIN
OUTPUT FILENAME:    /space2/fullVlbit/vlbit_1.difx

# CONFIGURATIONS ###!
NUM CONFIGURATIONS: 1
CONFIG NAME:        sess113.C512_ita
INT TIME (SEC):     2.000000
SUBINT NANOSECONDS: 80000000
GUARD NANOSECONDS:  254
FRINGE ROTN ORDER:  1
```

```
ARRAY STRIDE LENGTH:8
XMAC STRIDE LENGTH: 128
NUM BUFFERED FFTS:  10
WRITE AUTOCORRS:    TRUE
PULSAR BINNING:     FALSE
PHASED ARRAY:       FALSE
DATASTREAM 0 INDEX: 0
DATASTREAM 1 INDEX: 1
DATASTREAM 2 INDEX: 2
BASELINE 0 INDEX:   0
BASELINE 1 INDEX:   1
BASELINE 2 INDEX:   2

# RULES ############!
NUM RULES:          1
RULE 0 SCAN ID:     No0010
RULE 0 CONFIG NAME: sess113.C512_ita

# FREQ TABLE #######!
FREQ ENTRIES:       8
FREQ (MHZ) 0:       4966.49000000000
BW (MHZ) 0:         8.00000000000
SIDEBAND 0:         L
NUM CHANNELS 0:     128
CHANS TO AVG 0:     1
OVERSAMPLE FAC. 0:  1
DECIMATION FAC. 0:  1
PHASE CALS 0 OUT:   2
PHASE CAL 0/0 INDEX:1
PHASE CAL 0/1 INDEX:6

[...]

# TELESCOPE TABLE ##!
TELESCOPE ENTRIES:  3
TELESCOPE NAME 0:   MC
CLOCK REF MJD 0:    56397.6041666667
CLOCK POLY ORDER 0: 1
@ ***** Clock poly coeff N: has units microsec / sec^N ***** @
CLOCK COEFF 0/0:    1.230000000000000e+01
CLOCK COEFF 0/1:    1.687000000000000e-12
```

```
TELESCOPE NAME 1:   ND
CLOCK REF MJD 1:    56397.6041666667
CLOCK POLY ORDER 1: 1
@ ***** Clock poly coeff N: has units microsec / sec^N ***** @
CLOCK COEFF 1/0:    0.000000000000000e+00
CLOCK COEFF 1/1:    -1.060000000000000e-12
TELESCOPE NAME 2:   NT
CLOCK REF MJD 2:    56397.6041666667
CLOCK POLY ORDER 2: 1
@ ***** Clock poly coeff N: has units microsec / sec^N ***** @
CLOCK COEFF 2/0:    1.680000000000000e+00
CLOCK COEFF 2/1:    -1.060000000000000e-12

# DATASTREAM TABLE #!
DATASTREAM ENTRIES: 3
DATA BUFFER FACTOR: 32
NUM DATA SEGMENTS:  8
TELESCOPE INDEX:    0
TSYS:               0.000000
DATA FORMAT:        MKIV
QUANTISATION BITS:  2
DATA FRAME SIZE:    160000
DATA SAMPLING:      REAL
DATA SOURCE:        FILE
FILTERBANK USED:    FALSE
PHASE CAL INT (MHZ):1
NUM RECORDED FREQS: 8
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
CLK OFFSET 1 (us):  0.000000
FREQ OFFSET 1 (Hz): 0.000000
NUM REC POLS 1:     2
REC FREQ INDEX 2:   2
CLK OFFSET 2 (us):  0.000000
FREQ OFFSET 2 (Hz): 0.000000

[...]
```

```
# BASELINE TABLE ###!
BASELINE ENTRIES:   3
D/STREAM A INDEX 0: 0
D/STREAM B INDEX 0: 1
NUM FREQS 0:        8
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

[...]

# DATA TABLE #######!
D/STREAM 0 FILES:   48
FILE 0/0:           /space1/vlbit/apr15_mc/apr15_mc_no0001
FILE 0/1:           /space1/vlbit/apr15_mc/apr15_mc_no0002
FILE 0/2:           /space1/vlbit/apr15_mc/apr15_mc_no0003
FILE 0/3:           /space1/vlbit/apr15_mc/apr15_mc_no0004
FILE 0/4:           /space1/vlbit/apr15_mc/apr15_mc_no0005
FILE 0/5:           /space1/vlbit/apr15_mc/apr15_mc_no0006

[...]
```

4.1.5 calcif2 preprocessing

Before starting the actual correlation there are are a few steps to take in order to provide all the files mpifxcorr needs and the one that is missing is the .im file regarding the geometric and atmospheric delay. The vlbi-mgr machine in the IRA cluster provides the necessary informations when on any of the nodes the *calcif2* program is called on a .calc file previously generated by *vex2difx*

Program CalcServer contains the Goddard Space Flight Center CALC package version 9.1, used to compute geometric delay models for VLBI applications. It is a repackaged version of the same source code that is used to compute models on the VLBA correlator. It is configured to run as a server. All of its interactions are via Remote Procedure Call (RPC) calls from other programs, such as calcif2, which could be running on the same or different computer. This program only needs to be started once on a given machine using the startCalcServer script [2]. It is set to start automatically on machine vlbi-mgr of the cluster. Environment variable CALC SERVER is set for the user who is running the correlation.

```
Start: startCalcServer
Test: checkCalcServer $CALC SERVER
Stop: killall CalcServer
```

calcif2 computes a 5th degree polynomial every 120 seconds (typically), very closely resembling the delay model generation used at the VLBA hardware correlator. These polynomials are then evaluated at each model point. This results in a tremendous speedup at negligible loss of accuracy. By default *calcif2* will call CALC three times for each model point and calculates more accurate u, v, w coordinates from delay measurements made over a small patch of the sky:
$(u, v, w) = (-c\frac{d\tau}{dl}, c\frac{d\tau}{dm}, c\tau)$
where l, m are angular coordinates (in radians) relative to the delay center on the sky, τ is the delay at the delay center and c is the speed of light.

It connects via RPC to a running copy of CalcServer which must be running on a computer called $CALC SERVER, or on the specified computer if the -s option is used. If the output files (specified in the .calc file) exist and are current (have newer modification times than the .calc file, then the files will not be recreated unless the force option is used [2].

The output file will be a .im Polynomial interferometer model file, to be used by *difx2fits*.

```
CALC SERVER:        vlbi-mgr
CALC PROGRAM:       536871744
CALC VERSION:       1
START YEAR:         2013
START MONTH:        4
START DAY:          15
START HOUR:         12
START MINUTE:       48
START SECOND:       25
POLYNOMIAL ORDER:   5
INTERVAL (SECS):    120
ABERRATION CORR:    EXACT
```

```
NUM TELESCOPES:     3
TELESCOPE 0 NAME:   MC
TELESCOPE 1 NAME:   ND
TELESCOPE 2 NAME:   NT
NUM SCANS:          1
SCAN 0 POINTING SRC:BLLAC
SCAN 0 NUM PHS CTRS:1
SCAN 0 PHS CTR 0 SRC:BLLAC
SCAN 0 NUM POLY:    4
SCAN 0 POLY 0 MJD:  56397
SCAN 0 POLY 0 SEC:  46080
SRC 0 ANT 0 DELAY (us): 1.258536758483225e+04   -7.969338513877258e-01

[...]
```

4.1.6 mpifxcorr

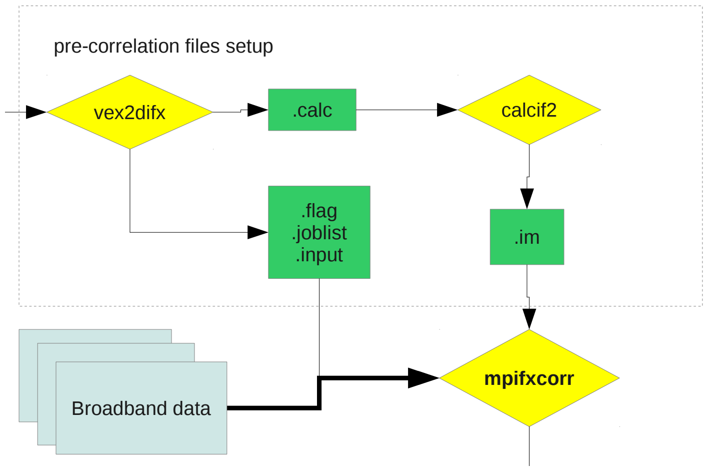


Figure 9: Core workflow of correlation process

The core of the DiFX software correlator is the program called mpifxcorr. This program is uses parallel computing to make correlation practical on a cluster of ordinary computers. This program runs on all the machines listed in the machines file that is passed to mpirun the program that starts mpifxcorr. It should be initiated from the cluster head node from within the project directory [2].

Usage: mpirun -np *nProcess* –bynode –hostfile [*otherOptions*] *machinesFile* mpifixcorr *input-File*

```
mpirun -machinefile machines -np 30 -bynode mpifxcorr vlbit_1.input
```

Another program might be launched afterwards that is *errormon2* that monitors the MPI processes on the cluster nodes to gain more informations.

To get an in-depth description of how mpifxcorr works there are available web pages describing the three blocks (FxManager, Datastream and Core) that compose it [16].

A benchmark example of the performance is correlating a 15 GB single scan of three antennas where on our 3 nodes cluster using 30 cores takes 120 seconds. That means the throughput is 125MB/sec processed.

4.2 DiFX output

The visibilities written by mpifxcorr are written to a directory with extension .difx. Typically there will be a single file in this directory, but it is possible that the output data will be split into multiple smaller files if the first output file gets too large or if correlation is continued from a point midway through correlation. These files contain visibility data records. Each record contains the visibility spectrum for one polarization of one baseband channel of one baseline for one integration time.

The following is an example output:

```
DIFX_56397_046105.s0000.b0000
PCAL_56397_046105_MC
PCAL_56397_046105_ND
PCAL_56397_046105_NT
```

The first file is a binary one, the other three contain the antenna code and a matrix of numbers.

Following the end-of-line mark for the last header row begins binary data in the form of (real, imaginary) pairs of 32-bit floating point numbers. The .input file parameter NUM CHANNELS indicates the number of complex values to expect. 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 [2].

This is the end point of the correlation, though it is the starting one depending on whether we choose to verify the correlated data for Astronomy or Geodesy and subsequently check the presence of fringes.

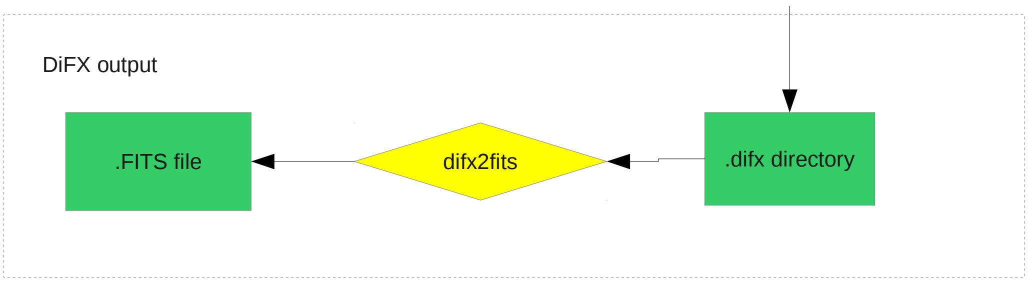


Figure 10: The DiFX output in heading to a FITS file

5 Fringe finding

After successfully correlating depending on the kind of experiment, either Geodetic or Astronomical, one can choose to follow two paths, the fourfit one can suit both geodetic and astronomical observations to find fringes, whereas obtaining a FITS file out of DiFX and following the AIPS pipeline is only suitable for astronomical experiments.

5.1 fourfit pipeline for Geodesy

Fourfit is part of the HOPS package that was developed at Haystack. The HOPS suite contains also a useful program *aedit* which will not be mentioned in this paper because it serves the purpose of managing huge amounts of experiments with data coming from several antennas. This is not going to be the case when planning local experiments.

5.1.1 difx2mark4

Program *difx2mark4* creates a Mk4 output fileset from the DIFX format visibilities created by mpifxcorr and several other files carrying information about the observation.

difx2mark4 should be invoked from the command line as follows:

difx2mark4 [*options*] [*baseFilename*]

baseFilenameX is the prefix of a jobfile to convert; it is OK to use the .difx filename instead

other options currently supported include:
-h or help : print usage information and exit
-v or verbose : increase verbosity of output; use twice or thrice to get even more
-e nnnn : put the output in a directory for experiment nnnn (defaults to 1234)
-d or difx : run on all .difx files in the directory, in which case the baseFilename is not necessary
override-version : runs even in the event of a mismatch of the difx2mark4 version and the datafile version
-r or raw : run in raw mode, which suppresses normalization

-k or ”keep-order’ : don’t sort antenna order into alphabetic

5.1.2 codes

The Mark4 correlation system represents stations with a single letter. Thus a 2-letter to 1-letter mapping is needed for each station. There is a default mapping (compiled in) which includes a number of stations and seven empty slots to be used as stations not in the table are encountered

A $\leftrightarrow$ Ai	B $\leftrightarrow$ Bd	C $\leftrightarrow$ Sh	D $\leftrightarrow$ 13
E $\leftrightarrow$ Wf	F $\leftrightarrow$ Eb	G $\leftrightarrow$ Gb	H $\leftrightarrow$ Ho
I $\leftrightarrow$ Ma	J $\leftrightarrow$ Cc	K $\leftrightarrow$ Kk	L $\leftrightarrow$ xx
M $\leftrightarrow$ Mc	N $\leftrightarrow$ Ny	O $\leftrightarrow$ Kb	P $\leftrightarrow$ Oh
Q $\leftrightarrow$ Tc	R $\leftrightarrow$ Zc	S $\leftrightarrow$ Nt	T $\leftrightarrow$ Ts
U $\leftrightarrow$ Ur	V $\leftrightarrow$ Wz	W $\leftrightarrow$ xx	X $\leftrightarrow$ On
Y $\leftrightarrow$ Yb	Z $\leftrightarrow$ Mh	a $\leftrightarrow$ Ap	b $\leftrightarrow$ Br
c $\leftrightarrow$ Cm	d $\leftrightarrow$ Cn	e $\leftrightarrow$ xx	f $\leftrightarrow$ Fd
g $\leftrightarrow$ xx	h $\leftrightarrow$ Hn	i $\leftrightarrow$ xx	j $\leftrightarrow$ Jc
k $\leftrightarrow$ Kp	l $\leftrightarrow$ La	m $\leftrightarrow$ Mk	n $\leftrightarrow$ Nl
o $\leftrightarrow$ Ov	p $\leftrightarrow$ Pt	q $\leftrightarrow$ Qb	r $\leftrightarrow$ Ro
s $\leftrightarrow$ Sc	t $\leftrightarrow$ Ti	u $\leftrightarrow$ Ur	v $\leftrightarrow$ Pv
w $\leftrightarrow$ Wb	x $\leftrightarrow$ xx	y $\leftrightarrow$ Y	z $\leftrightarrow$ xx

Table 3: HOPS Default Station Assignments

As each station not in the table is found, an xx entry in the table above is assigned to that station and the corresponding single letter code is used thereafter. If you have more than seven stations not in this table, or if you wish to assign the single letters that will be used for each stations, you must supply a file with new mappings and inform difx2mark4 via an environment variable HOPS STATION CODE which should be the (local or absolute) path to a file with the new one-letter $\leftrightarrow$ two-letter pairings. E.g.
A Aa
B Bb
C Cc
or equivalently
Aa A
Bb B
Cc C

Note that difx2mark4 enforces a one-to-one mapping, as bad things will happen if different stations were to be given the same one-letter identifier. (The hops software has no way to tell that this has happened, and difx2mark4 does not currently have any safeguard against this.)

So if you wish to reassign one of the existing assignments, you must first make it an open slot (i.e. assign it to xx) and then assign it.

5.1.3 fourfit

Fourfit searches the data represented by the root and corellation files for fringes, writing the results of the search to files of type fringe.

```
fourfit pt c control 1234
```

The command displays on X screen all the fringes found in the experiment scans after the DifX output is converted in Mark4 format. This is done passing the 1234 directory that contains all the scans correlated. On the IRA cluster a virtual PDF printer has been set up in order to get pdfs other that the default postscript files. The option hardcopy sends the fourfit output to the default printer.

```
fourfit pt c control -d hardcopy 1234
```

The control file typically used in fourfit should include the following lines referring to the phase cal mode and the width of the single band, multi band and delay rate search window expressed in microseconds.

```
pc_mode normal #pcal applied
sb_win -256.0 256.0 mb_win -2.0 2.0 dr_win -30.e-4 30.e-5
```

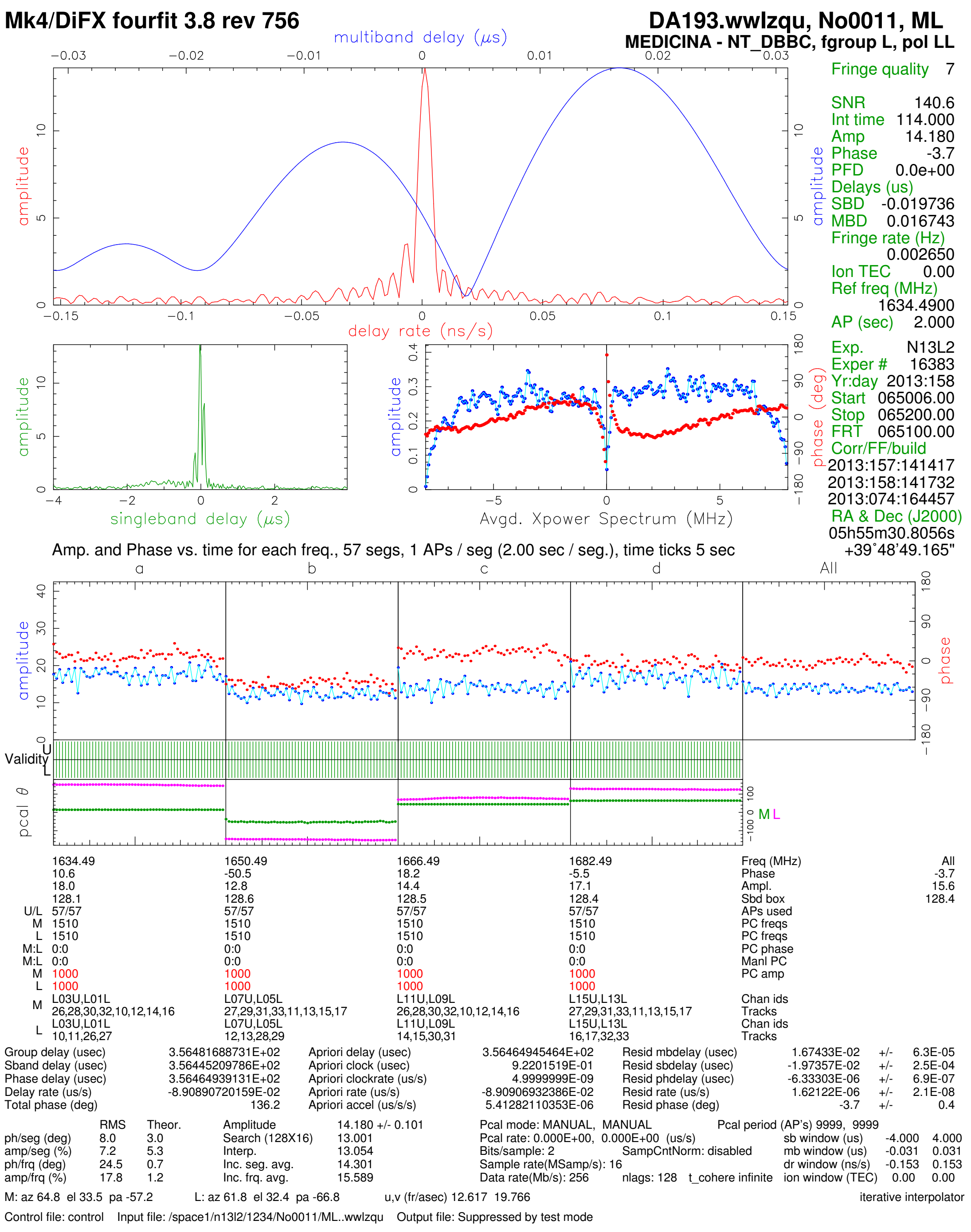
The most important values to check on a four fit output are the signal-noise ratio (SNR), the higher the better, and the single band delay (SBD) In the example on Figure 11 the fringe quality is set at 7 on a scale of 10 and the single band delay has an acceptable value. If the SBD is not approximating to 0 it is a sign the clock values have to be corrected in order to center the fringe and get a more accurate result. In fact an iterative process goes on while correlating: it is best suggested to correlate sources that are catalogued as calibrators and once fixed the residual delay, try to correlate all the remaining sources in an experiment.

The a, b, c, d columns correspond to the four sky frequencies observed (1634.49, 1650.49, 1666.49 and 1682.49 MHz) in L band.

5.1.4 mark4 data format

A mk4 fileset consists of a set of files all having the same 6 character alphabetic suffix, which is generated automatically based upon the current time, and which serves to keep filesets easily associated. Within the fileset there will be one ASCII pseudo-vex root file, named <source>.suffix. There will be one binary type 1 file for each correlated baseline, named e.g. QZ..suffix for stations Q and Z. There will also be a binary type 3 file for each station, named e.g. Q..suffix for some station Q.

This is the usual content of the default 1234 directory that comes out of *difx2mark4*.



```
Mk4/DiFX fourfit 3.8 rev 756                         DA193.wwlzqu, No0011, ML
                                      MEDICINA - NT_DBBC, fgroup L, pol LL

Fringe quality   7

SNR          140.6
Int time   114.000
Amp         14.180
Phase         -3.7
PFD        0.0e+00
Delays (us)
SBD      -0.019736
MBD       0.016743
Fringe rate (Hz)
          0.002650
Ion TEC       0.00
Ref freq (MHz)
         1634.4900
AP (sec)     2.000

Exp.         N13L2
Exper #      16383
Yr:day    2013:158
Start    065006.00
Stop     065200.00
FRT      065100.00
Corr/FF/build
2013:157:141417
2013:158:141732
2013:074:164457
RA & Dec (J2000)
05h55m30.8056s
+39°48'49.165"

Amp. and Phase vs. time for each freq., 57 segs, 1 APs / seg (2.00 sec / seg.), time ticks 5 sec

     1634.49                  1650.49                  1666.49                  1682.49                  Freq (MHz)       All
     10.6                     -50.5                    18.2                     -5.5                     Phase           -3.7
     18.0                     12.8                     14.4                     17.1                     Ampl.           15.6
     128.1                    128.6                    128.5                    128.4                    Sbd box        128.4
U/L  57/57                    57/57                    57/57                    57/57                    APs used
  M  1510                     1510                     1510                     1510                     PC freqs
  L  1510                     1510                     1510                     1510                     PC freqs
M:L  0:0                      0:0                      0:0                      0:0                      PC phase
M:L  0:0                      0:0                      0:0                      0:0                      Manl PC
  M  1000                     1000                     1000                     1000                     PC amp
  L  1000                     1000                     1000                     1000
     L03U,L01L                L07U,L05L                L11U,L09L                L15U,L13L                Chan ids
  M  26,28,30,32,10,12,14,16  27,29,31,33,11,13,15,17  26,28,30,32,10,12,14,16  27,29,31,33,11,13,15,17  Tracks
     L03U,L01L                L07U,L05L                L11U,L09L                L15U,L13L                Chan ids
  L  10,11,26,27              12,13,28,29              14,15,30,31              16,17,32,33              Tracks
Group delay (usec)        3.56481688731E+02  Apriori delay (usec)       3.56464945464E+02  Resid mbdelay (usec)   1.67433E-02  +/-  6.3E-05
Sband delay (usec)        3.56445209786E+02  Apriori clock (usec)          9.2201519E-01  Resid sbdelay (usec)  -1.97357E-02  +/-  2.5E-04
Phase delay (usec)        3.56464939131E+02  Apriori clockrate (us/s)      4.9999999E-09  Resid phdelay (usec)  -6.33303E-06  +/-  6.9E-07
Delay rate (us/s)        -8.90890720159E-02  Apriori rate (us/s)       -8.90906932386E-02  Resid rate (us/s)      1.62122E-06  +/-  2.1E-08
Total phase (deg)                     136.2  Apriori accel (us/s/s)     5.41282110353E-06  Resid phase (deg)             -3.7  +/-      0.4

               RMS    Theor.    Amplitude         14.180 +/- 0.101    Pcal mode: MANUAL,  MANUAL          Pcal period (AP's) 9999,  9999
ph/seg (deg)   8.0    3.0       Search (128X16)   13.001              Pcal rate: 0.000E+00,  0.000E+00  (us/s)      sb window (us)   -4.000   4.000
amp/seg (%)    7.2    5.3       Interp.           13.054              Bits/sample: 2        SampCntNorm: disabled   mb window (us)   -0.031   0.031
ph/frq (deg)   24.5   0.7       Inc. seg. avg.    14.301              Sample rate(MSamp/s): 16                      dr window (ns/s) -0.153   0.153
amp/frq (%)    17.8   1.2       Inc. frq. avg.    15.589              Data rate(Mb/s): 256   nlags: 128  t_cohere infinite  ion window (TEC)  0.00  0.00
M: az 64.8  el 33.5  pa -57.2      L: az 61.8  el 32.4  pa -66.8      u,v (fr/asec) 12.617  19.766                      iterative interpolator
Control file: control    Input file: /space1/n13l2/1234/No0011/ML..wwlzqu    Output file: Suppressed by test mode
```

Figure 11: A fourfit output example involving Medicina station and Noto dBBC

5.1.5 dbedit

The last step to produce a usable output for geodesy is to prepare the files for database publishing.

First, dbedit parses control file.

Then it reads so-called input media(s), what can be either a database or a set of databases, or correlator output or both.

If the input media is a correlator output dbedit creates a listing of directory tree which contains FOURFIT-supplied files: files of extent 52 in the case of Mark-3 correlator output and files of type 2xxx in the case of Mark-4 correlator output. Each input file corresponds to the fringe output for one observation. In general there may be several fringe outputs for the same observation. This list is saved in the temporary directory. Then dbedit reads all input files from that list. It extracts information from these files, makes some necessary transformations. Dbedit puts all extracted information related to the observation in the intermediary record.

If the input media is the database then the database is read. The database file is opened, table of contents is extracted. If the keyword INPUT LCODE is ALL or this keyword is not specified, then all lcodes are read. If keyword INPUT LCODE is specifies the list of permitted lcodes, then all lcodes which both saved in the database and specified in the INPUT LCODE lists are read. Dbedit puts all extracted information related to the observation in the intermediary record. If the file with unwanted observations is specified (keyword NOINPUT OBSER) then each observation which passed through all other filters is chacked agains the list of unwanted observation. If it matches to the observation specified inthe list it is not considered any more.

Then it applies input filters to the intermediary records of the observation if specified. If the record passes through the input filter it is written down in the temporarily file of direct access and a list of keys for each observation is created.

After that the keys are sorted in time order.

The duplicates are eliminated: records which correspond to the same observation [14].

5.2 AIPS pipeline for Astronomy

5.2.1 difx2fits

Program difx2fits creates a FITS output file from the DIFX (also known as SWIN) format visibilities created by mpifxcorr and several other files carrying information about the observation. When run, difx2fits requires the following files to be present:

baseFilename.difx/

As the visibility file (.difx) is read, any records that are all zero are omitted. The number count of these dropped records is reported as invalid records when difx2fits finishes writing the UV table. With difx2fits versions since 2.0 multiple correlator output files can be combined into a single destination FITS file.

```
difx2fits --difx
```

```
Writing FITS tables:
  Header                    2880 bytes
  AG -- array geometry      5888 bytes
  SU -- source              12360 bytes
  AN -- antenna             8004 bytes
  FR -- frequency           5624 bytes
  ML -- model               101724 bytes
  CT -- correlator (eop)    10810 bytes
  MC -- model components    22710 bytes
  SO -- spacecraft orbit    0 bytes
  GD -- pulsar gate duty    0 bytes
  GM -- pulsar gate model   0 bytes
  UV -- visibility

  Visibility files:
    JobId 0 file 1/1 : /space2/ivmcorr/ivm_1.difx/DIFX_56504_027000.s0000.b0000

    JobId 0 done.
      0 invalid records dropped
      0 flagged records dropped
      0 all zero records dropped
      0 negative weight records
      0 scan boundary records dropped
      0 out-of-time-range records dropped
      2604 records written
                            10969072 bytes
  FL -- flag                0 bytes
  TS -- system temperature  6528 bytes
  PH -- phase cal
    Station pcal file not found. No station pcal or cable cal measurements available
    No Tones extracted by DiFX
    No PC table will be written
                            0 bytes
  WR -- weather             0 bytes
  GN -- gain curve
    GAIN_CURVE_PATH not set; skipping GN table.
                            0 bytes
                            -----
  Total                     11145600 bytes

Conversion successful
```

```
1 of 1 jobs converted to 1 FITS files
```

5.2.2 AIPS setup

The Astronomical Image Processing System is a software package for calibration, data analysis, image display, plotting, and a variety of ancillary tasks on Astronomical Data. It comes from the National Radio Astronomy Observatory.

AIPS is installed at IRA on the h1 machine in /IRASOFT/AIPS. It can be found on most workstations already mounted and can quickly be launched with

```
aips tv=local
```

to view plots onto the screen one has to select the 7 option when prompted:

```
START_AIPS: Will use or start first available Unix Socket based TV

You have a choice of 7 printers.  These are:

    No. [ type  ] Description
-------------------------------------------------------------
     1. [     PS] Laser Computer Room-306 (DOUBLE pag)
     2. [PS-CMYK] Color Laser IRA Room-306 (DOUBLE)
     3. [     PS] Net-Laser HPPS Room-412
     4. [     PS] Laser HP Room-520
     5. [     PS] Laser HP-LaserJet-5M Room-515
     6. [     PS] Laser HP Quarto Piano
     7. [PREVIEW] Screen PS preview
-------------------------------------------------------------

START_AIPS: Enter your choice, or the word QUIT [default is 1]:
```

then choose an id number

```
AIPS 2: Enter user ID number
?
```

Given the assigned id number AIPS will be ready when the prompt displays

```
AIPS 2:                          31DEC13 AIPS:
AIPS 2:      Copyright (C) 1995-2012 Associated Universities, Inc.
AIPS 2:            AIPS comes with ABSOLUTELY NO WARRANTY;
AIPS 2:                 for details, type HELP GNUGPL
AIPS 2: This is free software, and you are welcome to redistribute it
AIPS 2: under certain conditions; type EXPLAIN GNUGPL for details.
AIPS 2: Previous session command-line history *not* recovered.
AIPS 2: TAB-key completions enabled, type HELP READLINE for details.
AIPS 2: Recovered POPS environment from last exit
>
```

AIPS is made up of tasks and procedures, the procedures summon different tasks and can speed up the analysis process. The procedures to load to reduce the VLBI data come from VLBAUTIL

```
>run vlbautil
```

5.2.3 Load and verify data

To load a FITS file into AIPS one has to run the following tasks:

```
>task 'fitld'

>datain '/FITSHOME/example.FITS

>go
```

Notice the syntax for the file path contains only a single apex at the beginning in order to make AIPS understand the path contains both uppercase and lowercase characters.

To display the files loaded:

```
>pca

AIPS 1: Catalog on disk  1
AIPS 1:  Cat Usid Mapname      Class   Seq  Pt     Last access      Stat
AIPS 1:    2  100 PROVA       .UVDATA.    1 UV 09-AUG-2013 17:09:39
```

The following step is to load the FITS file in memory getting the catalogue number identifying the FITS file to perform the subsequent tasks:

```
>getn 2
```

To check information about the antennas and the scans in a FITS data set:

```
>docrt 1
>vlbasumm

Scan  Source      Qual   Calcode Sub   Timerange          FrqID   START VIS  END VIS
1 PSNJ0136        : 0000           1  0/08:02:01 -   0/08:05:59     1       1        352
```

```
2 J0129+1446      : 0000            1  0/08:06:09 -   0/08:07:59     1     353           520
3 PSNJ0136        : 0000            1  0/08:08:09 -   0/08:12:59     1     521           958
4 J0129+1446      : 0000            1  0/08:13:31 -   0/08:15:29     1     959          1130
5 PSNJ0136        : 0000            1  0/08:15:39 -   0/08:20:29     1    1131          1568
6 J0129+1446      : 0000            1  0/08:20:39 -   0/08:22:29     1    1569          1736
7 PSNJ0136        : 0000            1  0/08:22:39 -   0/08:27:29     1    1737          2174

 [...]
```

Another important tool to check the data is imh that reads the FITS header. The fundamental extension files version to check in the list displayed are the solutions table (SN) and the calibration table (CL) values. They each show a progressive number after the calibrations are determined and appended to the FITS file.

```
>imh
AIPS 1: Image=MULTI     (UV)         Filename=PROVA       .UVDATA.   1
AIPS 1: Telescope=VLBA               Receiver=VLBA
AIPS 1: Observer=IVM                 User #=  100
AIPS 1: Observ. date=31-JUL-2013     Map date=09-AUG-2013
AIPS 1: # visibilities     13556     Sort order  TB
AIPS 1: Rand axes: UU-L-SIN  VV-L-SIN  WW-L-SIN  TIME1  BASELINE
AIPS 1:            SOURCE  FREQSEL  INTTIM  GATEID  CORR-ID
AIPS 1: ----------------------------------------------------------------
AIPS 1: Type    Pixels   Coord value     at Pixel     Coord incr   Rotat
AIPS 1: COMPLEX      3   1.0000000E+00       1.00  1.0000000E+00    0.00
AIPS 1: STOKES       2  -1.0000000E+00       1.00 -1.0000000E+00    0.00
AIPS 1: FREQ        32   6.6605356E+09       0.62  6.2500000E+04    0.00
AIPS 1: IF           8   1.0000000E+00       1.00  1.0000000E+00    0.00
AIPS 1: RA           1    00 00 00.000       1.00       0.000000    0.00
AIPS 1: DEC          1    00 00 00.000       1.00       0.000000    0.00
AIPS 1: ----------------------------------------------------------------
AIPS 1: Coordinate equinox 2000.00
AIPS 1: Maximum version number of extension files of type HI is   1
AIPS 1: Maximum version number of extension files of type PL is   1
AIPS 1: Maximum version number of extension files of type AT is   1
AIPS 1: Maximum version number of extension files of type SN is   1
AIPS 1: Maximum version number of extension files of type IM is   1
AIPS 1: Maximum version number of extension files of type CT is   1
AIPS 1: Maximum version number of extension files of type MC is   1
AIPS 1: Maximum version number of extension files of type NX is   1
AIPS 1: Maximum version number of extension files of type CL is   1
AIPS 1: Maximum version number of extension files of type FQ is   1
AIPS 1: Maximum version number of extension files of type AN is   1
AIPS 1: Maximum version number of extension files of type SU is   1
AIPS 1: Maximum version number of extension files of type CQ is   1
AIPS 1: Maximum version number of extension files of type TY is   1
AIPS 1: Keyword = 'OLDRFQ  '  value =  6.66252000D+09
```

5.2.4 Plot cross power spectrum

In order to check the calibration this procedure should be run using the calibration table number shown in vlbasumm through parameter GAINUSE. Then to show the plots on the screen prameter DOTV must be set to 1.

Procedure VLBACRPL relies most on POSSM task.

```
>inp vlbacrpl
AIPS 1: VLBACRPL: Procedure to plot cross power spectrum
AIPS 1: Adverbs     Values                 Comments
AIPS 1: ----------------------------------------------------------------
AIPS 1: INNAME     'PROVA'                 Input UV file name (name)
AIPS 1: INCLASS    'UVDATA'                Input UV file name (class)
AIPS 1: INSEQ         1                    Input UV file name (seq. #)
AIPS 1: INDISK        1                    Input UV file disk unit #
AIPS 1: SOURCES    *all ' '                Sources to plot
AIPS 1: TIMERANG   *all 0                  Time range of a calibrator
AIPS 1:                                     1-4 = start day,hr,min,sec
AIPS 1:                                     5-8 = end   day,hr,min,sec
AIPS 1: SUBARRAY      0                    Subarray, 0=>1
AIPS 1: REFANT        0                    Reference antenna; 0=> 1
AIPS 1: STOKES     ' '                     Stokes' parameters to plot
AIPS 1: GAINUSE       0                    CL table to use; 0=>highest
AIPS 1:                                      < 0 => no cal. applied
AIPS 1: SOLINT        1                    >0 defines averaging time
AIPS 1:                                    for each plot.  If -1 and
AIPS 1:                                    an NX table exists, will
AIPS 1:                                    do scan averages.
AIPS 1: DOTV          1                    =1 plot to TV, <=0 make plot
AIPS 1:                                      file
```

An example of the workflow looks like this:

```
>task 'vlbacrpl'

>gainuse 1
```

```
>dotv 1

>go vlbacrpl
```

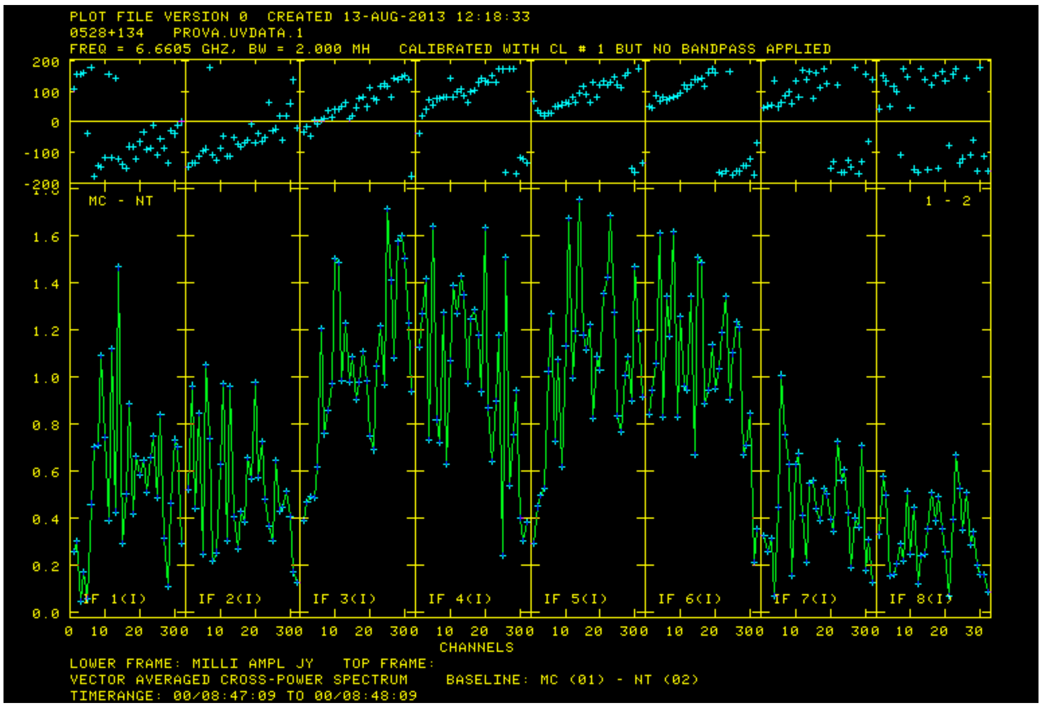


Figure 12: AIPS plot displaying non-calibrated data

5.2.5 Manually correct instrumental phases

VLBAMPCL is a procedure that creates a CL table which corrects the instrumental phases and delays using task FRING. It will run FRING twice if there are no scans that have good fringes on all antennas.

This part is about selecting the time range of typically the best scan available (previously checked in VLBACRPL procedure) for a source in order to get a new solution table (SN) to merge into a new calibration table (CL) to be applied when trying to find fringes.

```
>inp vlbampcl
AIPS 1: VLBAMPCL: Procedure to manually correct instrumental phases
AIPS 1: Adverbs      Values                      Comments
AIPS 1: ----------------------------------------------------------------
AIPS 1:                                          You must type RUN VLBAUTIL
AIPS 1:                                          to see all the inputs to
AIPS 1:                                          this procedure.
AIPS 1: INNAME      'PROVA'                      Input UV file name (name)
AIPS 1: INCLASS    'UVDATA'                Input UV file name (class)
AIPS 1: INSEQ         1                    Input UV file name (seq. #)
AIPS 1: INDISK        1                    Input UV file disk unit #
AIPS 1: TIMERANG   *all 0                  Time range of a calibrator
AIPS 1:                                     1-4 = start day,hr,min,sec
AIPS 1:                                     5-8 = end   day,hr,min,sec
AIPS 1:                                    MUST BE SPECIFIED
AIPS 1: BCHAN         1                    Lowest channel number 0=>all
AIPS 1: ECHAN         0                    Highest channel number
AIPS 1: REFANT        0                    Ref. antenna, 0=>the first
AIPS 1: SUBARRAY      0                    Subarray;  0 => all.
AIPS 1: CALSOUR    *all ' '                Calibrator source name;
AIPS 1:                                    blank => any
AIPS 1: GAINUSE       0                    CL table to use; 0=>highest
AIPS 1:                                    ADVERBS IF 2 SCANS MUST BE
AIPS 1:                                    USED TO CORRECT ALL ANTENNAS:
AIPS 1: OPCODE     ' '                     'CALP' if you are using 2
AIPS 1:                                    scans, otherwise ''
AIPS 1: TIME2      *all 0                  Time range of a calibrator
AIPS 1:                                     1-4 = start day,hr,min,sec
AIPS 1:                                     5-8 = end   day,hr,min,sec
AIPS 1: SOURCES    *all ' '                Calibrator source name for
AIPS 1:                                    TIME2 scan; blank => any
AIPS 1: ANTENNAS   *all 0                  antennas for which manual
AIPS 1:                                    phase corrections should be
AIPS 1:                                    obtained for the scan in
AIPS 1:                                    TIME2 (i.e. that cannot be
AIPS 1:                                    corrected with TIMERANG).
```

An example of this procedure:

```
>task 'vlbampcl'

>sources '0528+134'

>timerang 0 11 18 38 0 11 21 28

>go
```

As seen in the log output the task involved are FRING and CLCAL and both a new solution table and a calibration table are created.

```
gaia.i> FRING1: Task FRING  (release of 31DEC13) begins
gaia.i> FRING1: UVGET: doing no flagging this time
gaia.i> FRING1: Set integration time to  2.000000 seconds, hope that is ok
gaia.i> FRING1: Selecting and calibrating the data
gaia.i> FRING1: Dividing data by source flux densities
gaia.i> FRING1: Determining solutions
gaia.i> FRING1: Writing SN table    3
gaia.i> FRING1: Time=   0/ 10 21 02, Polarization = 1
gaia.i> FRING1: QINIT: did a GET  of      6268 Kwords, OFF        -59066402
gaia.i> FRING1: Time=   0/ 10 21 02, Polarization = 2
gaia.i> FRING1: Found       32 good solutions
gaia.i> FRING1: Adjusting solutions to a common reference antenna
gaia.i> FRING1: Appears to have ended successfully
gaia.i> FRING1: gaia 31DEC13 TST: Cpu=      3.4  Real=      5  IO=        10
gaia.i> CLCAL1: Task CLCAL  (release of 31DEC13) begins
gaia.i> CLCAL1: Using interpolation mode 2PT
gaia.i> CLCAL1: Processing SN table    3
gaia.i> CLCAL1: SNMRG: Merging SN table
gaia.i> CLCAL1: SNMRG: Write       2 merged records from       2 input records
gaia.i> CLCAL1: SN2CL: Applying SN tables to CL table   1, writing CL table  4
gaia.i> CLCAL1: Appears to have ended successfully
gaia.i> CLCAL1: gaia 31DEC13 TST: Cpu=      0.2  Real=      2  IO=         5
```

5.2.6 Fringe fit and calibrate data

Now we can try to find fringes resetting the GAINUSE parameter to 0 to apply the most recent calibration table and setting the solution interval to a value of 1 minute.

```
>task 'vlbafrng'

>gainuse 0

>solint 1

> go vlbafrng

>inp vlbafrng
AIPS 1: VLBAFRNG: Procedure to fringe fit and calibrate data
AIPS 1: Adverbs     Values                 Comments
AIPS 1: ----------------------------------------------------------------
AIPS 1:                                    Input uv data.
AIPS 1: INNAME     'PROVA'                    UV file name (name)
AIPS 1: INCLASS    'UVDATA'                   UV file name (class)
AIPS 1: INSEQ         1                        UV file name (seq. #)
AIPS 1: INDISK        1                        UV file disk drive #
AIPS 1: CALSOUR    *all ' '                 Source list to fringe fit
AIPS 1: TIMERANG   *all 0                   Time range to use.
AIPS 1: BCHAN         1                     Lowest channel number 0=>all
AIPS 1:                                     FOR SPEC. LINE EXP. ONLY
AIPS 1: ECHAN         0                     Highest channel number
AIPS 1:                                     FOR SPEC. LINE EXP. ONLY
AIPS 1: GAINUSE       0                     CL table to apply.
AIPS 1: REFANT        0                     Reference antenna
AIPS 1: SUBARRAY      0                     Subarray, 0=>all
AIPS 1: SEARCH     *all 0                   Prioritized reference antenna
AIPS 1:                                     list - supplements REFANT
AIPS 1: SOLINT        1                     Solution interval (min)
AIPS 1:                                     0 => 10 min
AIPS 1: DPARM      *all 0                   for strong sources, it is
AIPS 1:                                     only necessary to set
AIPS 1:                                     DPARM(4), DPARM(7) and for
AIPS 1:                                     spec line exp DPARM(8).
AIPS 1:                                        4=int. time (sec)
AIPS 1:                                         0 => min. found in data
AIPS 1:                                        7 >0 => don't rereference
AIPS 1:                                          phase; this should be 1
AIPS 1:                                          for polarization exp
AIPS 1:                                        8 > 0 => activate zero'ing
AIPS 1:                                                  options
AIPS 1:                                     for rest see HELP VLBAFRNG
AIPS 1: SOURCES    *all ' '                 Source list to calibrate: If
AIPS 1:                                     sources are listed they are
AIPS 1:                                     each calibrated with separate
AIPS 1:                                     runs of CLCAL
AIPS 1: INTERPOL   ' '                      Interpolation function:
AIPS 1:                                     '2PT','SIMP','AMBG','CUBE',
AIPS 1:                                     'SELF'
AIPS 1: BADDISK    *all 0                   Disk no. not to use for
AIPS 1:                                        scratch files.
```

If solutions are found, a new run of VLBACRPL should show phases aligned to 0 as we have now removed time dependencies. During the run of this procedure in the log one should check how many good solutions were found over the total. If it is a reasonable number then the correlation can be considered successful and the experiment worth further evaluation by Astronomers.

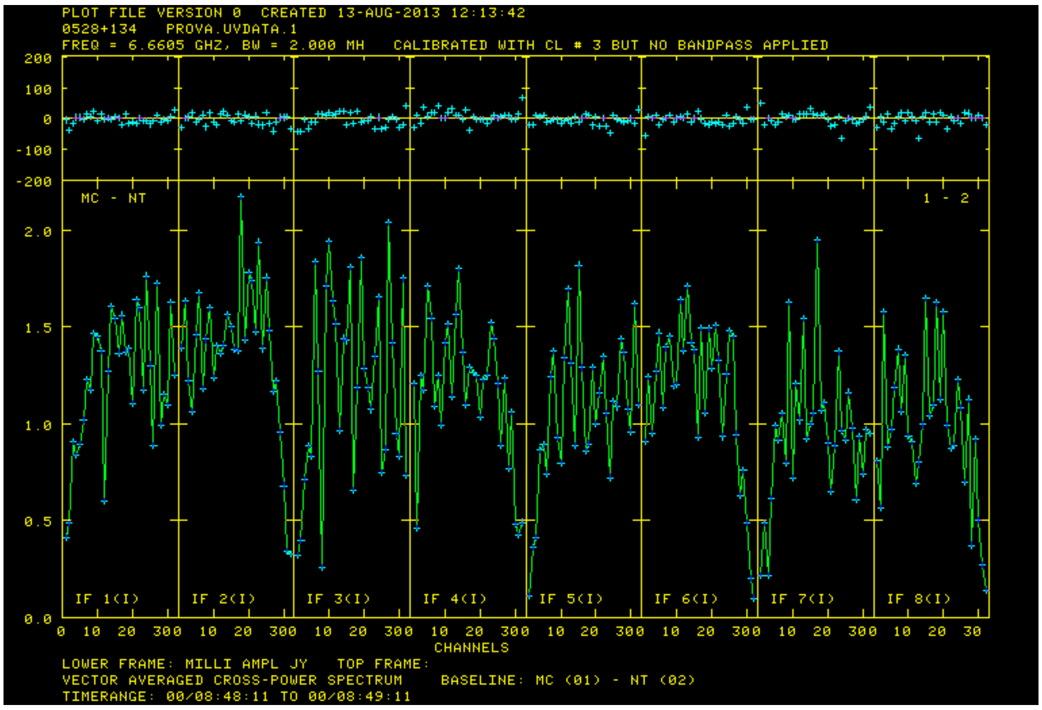


Figure 13: AIPS plot displaying now fitted phases calibrated with a new CL

6 Conclusions

The most challenging task producing this paper was gathering over a year's time all the necessary information to perform a correlation from A to Z, that means starting from installing DiFX and its ancillary software to produce something valuable to the end users, Geodesists and Astronomers alike.

Sometimes informations like the bitstream assignments or the clock signs were thought to be known and obvious by the insiders, causing many problems for us who were first making an attempt to get into this subject. In many senses this was the spur that led us to fill the gaps we found on our way and produce this manual.

Although some in-depth and theoretical parts may have been underestimated or neglected we hope to have produced a valuable guide to somebody who tries to approach or set up a correlation workflow for the first time.

References

[1] Walter Brisken. http://www.vlba.nrao.edu/memos/sensi/sensimemo23.pdf.

[2] Walter Brisken. *A Guide to the DiFX Software Correlator*, 1.5.3 edition, April 2010.

[3] CalTech. ftp://ftp.astro.caltech.edu/pub/pgplot/pgplot5.2.tar.gz.

[4] CalTech. http://www.astro.caltech.edu/t̃jp/pgplot/install-unix.html.

[5] CIRA. http://cira.ivec.org/dokuwiki/doku.php/difx/vex2difx.

[6] John Gipson. http://lupus.gsfc.nasa.gov/files_user_manuals/sked/sked.pdf.

[7] Hat-Lab. http://www.hat-lab.com/hatlab/products/dbbc.

[8] IERS. http://www.iers.org/.

[9] Intel. http://software.intel.com/en-us/intel-ipp.

[10] IRA-INAF. ftp://ftp.ira.inaf.it/pub/gps/.

[11] MIT. ftp://gemini.haystack.mit.edu/pub/hops/readme-3.7.txt.

[12] MIT. http://www.haystack.mit.edu/tech/vlbi/hops.html.

[13] NASA. http://lupus.gsfc.nasa.gov/fsdoc/fshome.html.

[14] Lonid Petrov. http://lacerta.gsfc.nasa.gov/mk5/help/dbedit_01.html.

[15] Matteo Stagni, Francesco Bedosti, and Mauro Nanni. Performance comparison of filesystems on raid for e-vlbi data. Technical Report 470/13, IRA - INAF, http://www.ira.inaf.it/Library/rapp-int/470-13.pdf, April 2013.

[16] Swinburne. http://cira.ivec.org/dokuwiki/doku.php/difx/mpifxcorr.

[17] Swinburne. http://groups.google.com/group/difx-users/.

[18] Swinburne. https://svn.atnf.csiro.au/trac/difx.

[19] Open MPI Team. http://www.open-mpi.org.

[20] Various. http://plplot.sourceforge.net.

[21] Harro Verkouter. http://www.radionet-eu.org/radionet3wiki/lib/exe/fetch.php?media=na:eratec:tog:tog-meeting-01:mark5c-tog-june2012.pdf.

[22] Craig Walker. http://www.aoc.nrao.edu/software/sched/.

[23] Alan Whitney. http://www.haystack.mit.edu/tech/vlbi/mark5/mark5_memos/039.pdf.