

Setting up DBBC for geodetic VLBI experiments

At the moment the Hb DBBC unit is set up under Windows XP. Operation under Linux (SuSE 10.3) is also possible with (beta version) software and FS support. This document describes the main tasks to set up the unit for a VLBI observation.

First obtain the latest firmware 'dbbc2.bit' and control software 'DBBC Control.exe' which will be temporarily made available on <ftp.mpifr-bonn.mpg.de/outgoing/p062gra/dbbc> . The file dbbc2.bit should be put in `c:\DBBC_CONF\FilesDBBC` the executable can go on Desktop.

The DBBC IF filters used are '1' which cover 500-1000 MHz and '2' which are for 100-500MHz. The DBBC has four FPGA boards, each of these provides 4 BBC channels, the IF channel 'A' can currently only provide data to the first board, therefore BBCs 1-4, channel B is for BBCs 5-8 and so on. This is important to note. It is a limit in the current firmware, not in hardware.

Two types of geo experiments are typically used: the 'narrowband' type (e.g. OHIG series) with X-band IF channels in the range 100-500 MHz, the 'R' series has 100-900 MHz. In the first case the IF channel 'B' uses filter '2' for BBCs 4-8, for the higher frequencies of the R series filter '1' must be used. Take as an example OHIG68:

- X-band in range 100-500 to IF input A1
- X-band in range 100-500 to IF input B1
- S-band in range 100-500 to IF input C1
- S-band in range 100-500 to IF input D1

To avoid having to set the frequencies and bandwidths by hand, they can be edited into the file 'C:\DBBC_CONF\dbbc_config_file.txt' as follows:



```
dbbc_config_file.txt - Blocco note
File Modifica Formato Visualizza ?
1 dbbc2.bit 130.99 4
1 dbbc2.bit 140.99 4
1 dbbc2.bit 170.99 4
1 dbbc2.bit 230.99 4
1 dbbc2.bit 340.99 4
1 dbbc2.bit 420.99 4
1 dbbc2.bit 470.99 4
1 dbbc2.bit 490.99 4
1 dbbc2.bit 192.99 4
1 dbbc2.bit 207.99 4
1 dbbc2.bit 217.99 4
1 dbbc2.bit 247.99 4
1 dbbc2.bit 267.99 4
1 dbbc2.bit 272.99 4
1 dbbc2.bit 600.80 4
1 dbbc2.bit 600.80 4
```

The commands to load the FPGA firmware are set in `C:\DBBC_CONF\configure.cmd`:

```
setMode -bs
setCable -port usb21 -baud 1200000
setPreference -pref UseHighz:TRUE
Identify
IdentifyMPM
assignFile -p 1 -file c:/DBBC_CONF/FilesDBBC/dbbc2.bit
assignFile -p 2 -file c:/DBBC_CONF/FilesDBBC/dbbc2.bit
assignFile -p 3 -file c:/DBBC_CONF/FilesDBBC/dbbc2.bit
assignFile -p 4 -file c:/DBBC_CONF/FilesDBBC/dbbc2.bit
program -p 1
program -p 2
program -p 3
program -p 4
quit
```

Connect external 10MHz and 1PPS. Start the control software 'DBBC Control.exe' (at least version dated 18.02.2010), reply to query if dbbc to be reconfigured with Y, then FPGAs will be programmed and frequencies set. Check that FPGAs load correctly, after the programming of each

card a message appears on the screen reporting whether programming has been successful or not. Watch this carefully. The FPGAs are programmed in 'highz' mode, this is meant to ensure that programming is successful even if they already contain code. Check that the same pattern of off and on LEDs is shown on all cards.

Sometimes this does not work, so it is often better to reset the FPGAs to a 'blank' state. This is done by switching power to the electronics off and then on (EL switch on back) before starting the control program. The software sets the ad9858 clock generator, BBC frequencies and bandwidth and should synchronize the 1pps of the FPGAs so they all flash together. Check that this is the case. Just to make sure, issue the command 'pps_sync'.

The IF channels should now be set up. Input level should be approximately -30dBm in 500MHz. Too much will overload the first IF amplifier, too little will not allow reaching the required levels. Input bandpasses should be flat. Slopes of more than a few dB will cause correlation losses. In the 'OHIG' case described above, the dbbcif commands (using 100-500MHz on all inputs) are:

```
dbbcifa=1,agc,2
dbbcifb=1,agc,2
dbbcifc=1,agc,2
dbbcifd=1,agc,2
```

After a few seconds, use the commands ddbcifa, dbbcifb etc. to read out the power level achieved. Should be at least 48000-50000 for all channels. The frequencies of the individual bbc channels can be checked with a dbbc call without parameters, for instance 'dbbc01'. The total power readings produced may not be reliable at the moment. An analogue monitor output is available, for instance with 'dbbcmon=b09u' to check with a spectrum analyser if familiar S-band interfering signals can be seen. In switch-on state the standard 'geo' track pattern for mk5b should be set. Make sure with 'dbbcform=geo'.

The next job is setup of the Mk5b unit. This **MUST** have its CPU synchronized to a reliable ntp server. Check this with a call to ntpq, using the command 'peers'. The offset to the ntp servers chosen should be in the range of a few milliseconds. The CPU time of the Mark5b unit is used to generate the time code of the data. *When Mark5C is deployed, the time code will be generated in the DBBC firmware.* The bit mask (typically 55555555 for 1-bit geo), bit rate and recording start/stop of the mark5b are controlled by the FS. Set the FS terminal type to 'mark5' or 'vlba5' in equip.ctl and skedf.ctl. This ensures that drudg option 11 can be used to generate the correct bit mask for the mark5b. The FS will complain that it cannot talk to the terminal and that its 'vsi4' commands have no effect, but this does not matter.

The internal 1PPS of the Mark5 must also be synchronized. The best way to do this is with the FS program 'fmset'. This will sync the mark5b and also show if mark5b time is the same as FS CPU time and FS internal time. All should be the same, and syncerr should be zero. Run fmset to check sync at intervals during the experiment. Run the FS command 'mark5=dot?' and put this command in the midob procedure. The last field reported by the 'dot?' command is time difference between the CPU second of the mark5b and the VSI 1PPS. This should generally be a few milliseconds only, reflecting the fact that ntp is OK, and DBBC is synced. It is recommended to have a counter (e.g. hp 53121A) between 1PPS out on the mark5b ('dotmon' on rear panel) and 1PPS from GPS. This is the equivalent of monitoring 'fmout' on the mark4 terminal. Data should be logged.

After a small amount of data has been recorded, write out a few Megabytes on mk5b system disk with the 'disk2file' command. Check the result with 'od -Ad -tx4' on the file you wrote. If the file written was taken from the beginning of the scan, the mark5b keyword should be in the first word, this is 'ABADDEED', followed by time codes and frame counters. Each header contains 4 words (16 bytes) and comes every 2500 words (10000 bytes). The data should be a random mess of hex digits. If you see a lot of zeros, something is badly wrong, for instance FPGA code not loaded, no input to DBBC.

```
0000000 abaddeed bead0000 18063003 0000013d
0000016 002ec3b8 82b3839c 431b03c9 003c0356
0000032 0331c0b9 c11a40ae 01bc03b3 c0fe8284
0000048 01a6c39c 011383e7 03b04122 431481b2 ..
```

It is also possible to list just the headers with a 'grep' on 'abaddeed':

```
0000000 abaddeed bead0000 18063003 0000013d
0010016 abaddeed bead0001 18063003 00060129
0020032 abaddeed bead0002 18063003 00120151
0030048 abaddeed bead0003 18063003 0018016d
```

The programs 'bstate' and 'vlbi2' which can be downloaded from

<ftp://web.haystack.edu/pub/mark5/B/util/>

are useful to check bit statistics and spectra. They are meant for 16-channel, 2-bit data, so the results are not correct for 1-bit data, but some indication of data health can be obtained. Note that to get correct bit statistics a subsequent version of the control program will be needed which is not yet available by ftp.