

# Starting an experiment using DBBC and Flexbuff

## Fieldsystem

Download all the necessary files.

## Modifying downloaded files

In order to create the station-specific files that can be run by the Field System(FS), one needs to run the *drudg* program from the *sched* directory.

## *drudg* on the Field System

Generating station-specific \*.prc and \*.snp files using *drudg*:

1. Go to the /usr2/sched directory (*cd sched*)
2. To start *drudg* type: *drudg sessionName.skd*, where the *sessionName.skd* is the sked file downloaded from the IVS ftp server
3. Program will ask for the station for which the output should be generated. For this type *On* (Onsala station).
4. Show/set equipment type - type: *11* - this option allows to change the type of equipment for which the SNAP file and procedure are generated. One needs to check using this option whether we are generating outputfiles for the proper equipment ( e.g. Flexbuff and DBBC2/3)
5. Make SNAP file (.SNP) - type: *3* (on a keyboard)
6. Make procedures (.PRC) - type: *12* (on a keyboard)  
For prompt *Enter TPI period in centiseconds (default is 1000, 0 for OFF)*  
press "enter"
7. Print summary of (.SNP) file - type: *5* - prints out the observing schedule
8. Print notes files (.TXT) - type: *51* - prints out the notes from the IVS if the *session Name.txt* has been downloaded along with the *sessionName.skd*
9. To exit *drudg* type: *0*

## Adding procedures for flexbuff (*No need any more drudg now has support for Flexbuff*):

If recording using flexbuff is planned, additional modification of drudged files needs to be done. We use some python scripts to replace the mk5 commands with flexbuff commands:

1. start *fila5cflex.py* with arguments *SESSION mk5b(recorder type)flex(flexbuff)* in *usr2/sched/* - this will store the data on flexbuff in the mk5b format
2. start *fila5cflex.py* with arguments *SESSION vdif(recorder type)flex(flexbuff)* in *usr2/sched/* - this will store the data on flexbuff in the vdif format
3. start *fila5cflex.py* with arguments *SESSION mk5b+vdif(recorder type)flex(flexbuff)* in *usr2/sched/* - this will store the data on flexbuff in mk5band vdif format. The recorder type defined before "+" is the primary one. In case of the secondary format, scans will be saved on flexbuff using "od" instead of "on" (e.g. "r1776 od 023-1700")

## Preparing for flexbuff recordings

Before an observation, the flexbuff intended for recording on should be checked for operation. This can be performed using the `check_flex.py` script, which checks whether a flexbuff is powered, running jive5ab, and informs the user how many disks are available for recording on. Once the flexbuff has been checked, the field system can be started, and it is then necessary to select the correct flexbuff to be used for recording on. This is achieved by running `finset`. Having selected a flexbuff to record on, you can then initiate a short test recording by using the `testrec.py` script to check that everything looks OK. This attempts to record around 10s of data and runs `scan_check` on the recording. In order for this to succeed, the field system needs to have initialised the flexbuff for the coming experiment (execute `sched_initi`, `setup01` or similar).

### Using `check_flex.py`

The `check_flex.py` script exists on all the field systems and can be used for diagnosing issues that may arise with flexbuffs. With no parameters, it will probe all known flexbuffs, looking for a jive5ab on the correct port for the field system computer the script is being run from. If successful, it reports the version of jive5ab running on each flexbuff, counts the number of disks available for recording. If any issues are detected, these are reported at the end, along with hints on what may be performed to fix the issues.

`check_flex.py` takes 2 possible parameters; `-fb <flexbuff>` and `-fs <field system>`, both of which can be supplied multiple times.

`-fb <flexbuff>`

Indicates a specific flexbuff you wish to investigate. Possible options are *kare*, *bogar*, and *skirner*. Without this parameter, all three flexbuffs will be probed.

`-fs <field system>`

Indicates the field system you wish to test from. Each field system uses a specific port for communicating with all flexbuffs. Without this parameter, only the communication port of the field system running `check_flex.py` will be used. By specifying this parameter one or more times, you can probe flexbuffs as if you were running `check_flex.py` on other field systems. A special option `-fs all` probes for all 4 field systems; *fold*, *rane*, *fulla*, and *freja*.

As an example,

```
check_flex.py -fb flexbuff1 -fb flexbuff2 -fs fs1 -fs fs2
```

checks jive5abs on both bogar and kare for both rane and fold.

To receive the datastreams from a DBBC it needs to listen on the correct network port and it does that with the program “jive5ab”.

”ssh flexbuff”

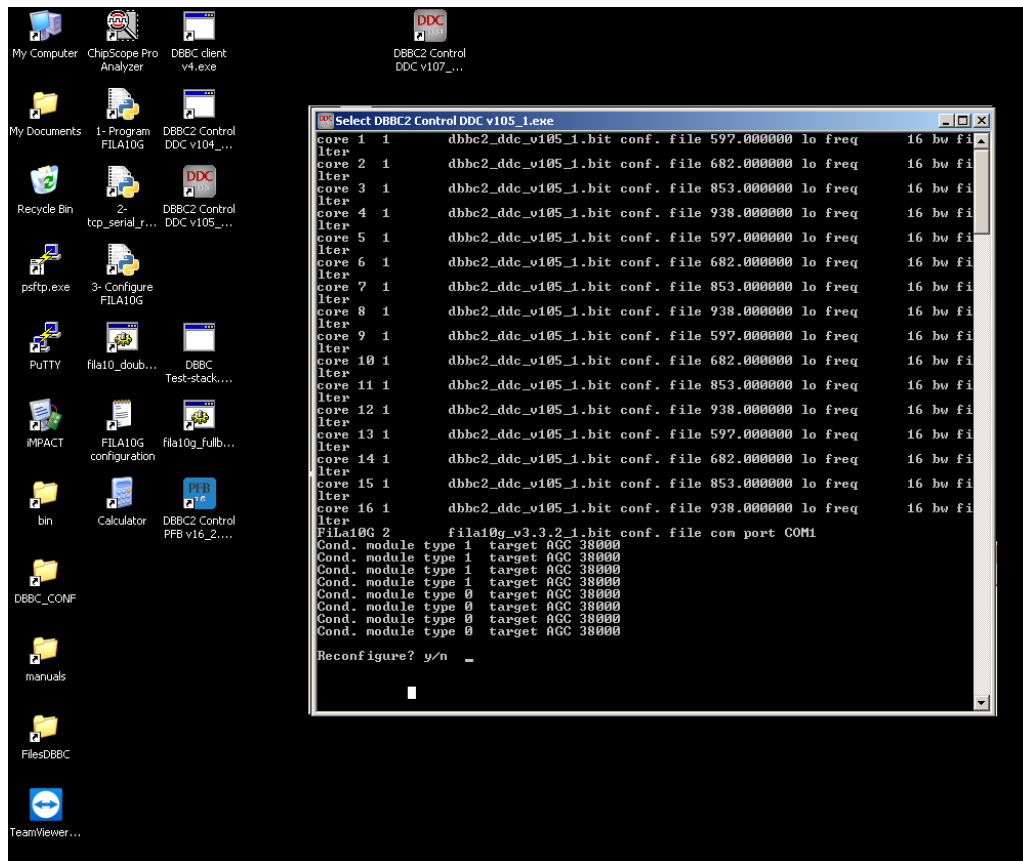
”StartJ5 mk5b -p32620” or ”StartJ5 vdif -p32620” to store the data flow in the mk5b+ or vdif format, respectively

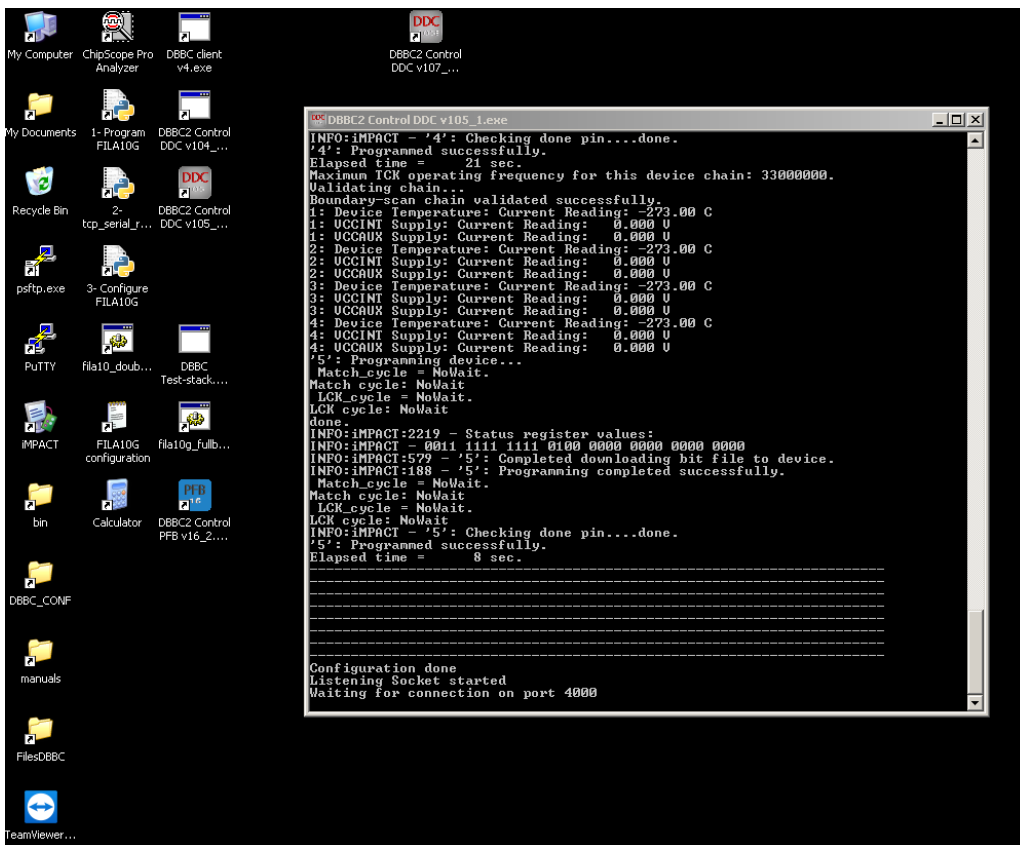
”StartJ5 mk5b -p32620” and ”StartJ5 vdif -p32620” to store the data flow in the mk5b+ and in the vdif format

## DBBC2/3 setup:

The DBBC2/3 are windows XP based systems that simply listens on a network port when it has configured the electronics.

Usually there is no need to start the DBBC2 (*DBBC2 control v105* executable ) since it is running all the time.  
If not just double click the executable and answer "Y"





Here the setup is complete and now it listens on port 4000.

## Fieldsystem

- . start schedule
- . type setupsx
- . sy=dbbcpath xs (selecting S/X for the DBBC)
- . type collect - displays values related to DBBC

Attenuation from the *collect* command:

```

2016.326.15:05:25.88/ifa/1,agc,2,38000,44,38133
2016.326.15:05:25.88/ifb/1,agc,1,38000,38,38103
2016.326.15:05:25.88/ifc/1,agc,2,38000,63,43332
2016.326.15:05:25.88/ifd/1,agc,2,38000,63,38192

```

. Bold is the target value you should aim for. Setting attenuation level (dB) to fit the target values is possible using *dbbcatt* command:

sy=dbbcatt 1/2 (S or X band) val (value of attenuation in dB (1-11) ) This will lower last values of each line.

Do clock offset reading.

Do a pointing check.

Do a cable delay reading.

### Start the session:

- . type *proc=xxx* - switch back to the observing procedure
- . type *log* - check whether the proper log file is used
- . type *schedule* - check whether the proper schedule file for station is used
- . check whether the time is synchronized with *fmset* (ctrl+shift + t)

. type *caltsys* - displays system temperatures of used channels

. type *cont* - starts observing procedure; antenna will start tracking the first source of the schedule and the measurements will start automatically in accordance to the schedule for the station

Verify the attenuator levels again with *collect* and *dbbcatt*

. type *mk5relink* - resets the tie - not mandatory if clock offsets are correct; behaves the same as *cont*

. type *sched\_initi* - initializes schedule and sets back the clock offsets; type this command if something is messed up with the clock offsets or Fila.

. Type *testrec.py* on fs in a new terminal window to run a trial recording test

. Turn on the bandpass plot– type “*gv bpass*” on FS in a new terminal window in order to turn on the creation of the plot with autocorrelation for each channel. This will be updated once a while during the experiment.

. type *checkmk5* to update the correlation plot.

## Controlling recordings on the flexbuff in different ways

Do a normal *scan\_check*?

Scan check output should look something like this:

```
2019.091.18:07:30.69/scan_check/?,vi9091_oe_091-1806,VDIF,128,2019y91d18h6m56.1460s,30.0396s,64.000000,-402976,8224
2019.091.18:07:30.69/mk5_status/status,0x00000001
2019.091.18:07:30.69/mk5/net_protocol? 0 : udpsnor : 67108864 : 268435456 : 8 ;
2019.091.18:07:30.69/mk5/mtu? 0 : 9000 ;
2019.091.18:07:30.69/mk5/rtime? 0 : 142237s : 146219GB : 38.5311% : VDIF : 2 : 0MHz : 8224Mbps ;
```

Login onto flexbuff and then type: *ls /mnt/d\*/sessionName\**

in terminal to check whether something has been transferred to the flexbuff. Check a size of the newly recorded files: “*vbs ls -lvt sessionName\**”

Testing data recording on *flexbuff* (*this is done automatic, it's added in “scan\_check”*) :

This is a script that takes 10 secs of data and transfers it to the flexbuff where it uses *m5spec* and *m5bstate* to generate a postscript tfile that's transferred back to the FS and displayed with Ghostview.

Execute *testrec.py* to carry out a test recording of data onto flexbuff, (it takes 10 seconds of data). Then check the displayed status of the code and look also into a login shell for the output from the field system.

## View of FS during session

