o

Mark6 VLBI Operator's Guide

Summary of steps involved in the observation

1. Get *.vex, *sch.ar files; create the setup file; move the file to /share/obs4/usr/vlbi/
2. Create snp file (for EVN use flexbuff; for HSA use Mark6)
3. Check enough disk space is available in Mark6 (In oper@mark6 df -kh). Delete old files to create space if there is a shortage. For fringe test, remove old files from /home/oper/data (cmd rm *)
4. 15 minutes before obs connect all cables, set up the total power detector
5. Start cima, user vlbi, your initial, line; select receiver; run setup
6. Adjust the attenuator in the total power detector to get 0.5 in the display
7. Vncviewer vlbis1:2 (123456)
8. In vlbis1: ssh oper@mark6 (naic305m); check jive5ab is running (ps -elf | grep jive)
9. Reboot RDBEs (/usr2/oper/bin/*rdbe2_reboot.sh* and *rdbe4_reboot.sh*)
10. **Wait till RDBEs reboot** (Check they have rebooted by issuing, for example, dbe_rms? From vlibish)
11. Run FS; check FS time is synched with UT time (if not follow the steps to synch it)
12. In FS, schedule=xxxar,#1 (xxar.snp is the snp file)
13. **Wait till RDBEs are initialized**
14. In vlbis1: run /usr2/oper RDBE2_AGC_AR.py and RDBE4_AGC_AR.py to set RMS to 20
15. In vlbis1: vlbish - run rdbe2, rdbe4/dbe_rms? To get the RMS values
16. In vlbis1: vlbish - run command rdbe2,rdbe4/dbe_ddc_quantize?
17. In vlbis1: run /usr2/oper/bin/ddc_quantize_adj.py to get threshold values
18. In vlbis1: vlbish - run command dbe_ddc_quantize=x:+th:zth,-th, for x=0-3
19. Check for sync err (vlbish dbe_dot?); if not synced in vlbish dbe_dot_set=;
20. Steps 14-18 can be done by running the files vlbi_init_monitor_rdbe2.py and vlbi_init_monitor_rdbe4.py in the folder /usr2/oper/bin
21. Wait for scan to start and check data is being recorded
22. In vlbis1: vlbish - run dbe_data_send? To check data is send from RDBE
23. In oper@mark6 vbs_ls -l6 | grep xxar* to check data is being written

Preparing for VLBI each month/semester/year
Review emails for messages from HSA/EVN concerning upcoming runs,
do technical reviews about whether our capabilities align with proposed projects,
coordinate VLBI scheduling with Hector.

Before Observation:

(A) Login to oper@vlbis1 (+p w d) and sftp the required files to /usr2/sched/ as described below. Usually a vncserver will be running on vlbis1. To connect to the vncserver use the

following command from a linux server. If it is not running see Appendix on how to create a vncserver on vlbis1

Vncviewer vlbis1:2 (+p w d 123456)
On a Mac, use the following command on the terminal:
>> open vnc://vlbis1.naic.edu:5902

If another account is used to download the required files then copy those files to vlbis1 using the command

Scp *files* oper@vlbis1:/usr2/sched/.

(1) For EVN Observations: get them from vlbeer

Commands to download
sftp evn@vlbeer.ira.inaf.it
morevlbeer

cd vlbi_arch
cd oct19 -- month and year (example oct 2019)
cd .latest

To download files for run starting with n19c*
get n19c***.vex
get n19c**sch.ar
get n19c**.sum
quit

(2) For HSA/VLBA Observations: get then from NRAO

Commands to download

ftp ftp.aoc.nrao.edu
vlbiobs
LUV2snif

cd astronomy
cd oct19 -- month and year (example oct 2019)
cd n19c** (project number)

To download files for run starting with n19c*
get n19c***.vex
get n19c**sch.ar
get n19c**.sum

quit

(B) Commands to print the *sch.ar (or *.sum) files on a Arecibo printer in a form that is readable is:
> enscript -r -Php0 ******sch.ar (replace “hp0” with any other printer-name, if appropriate)

(D) Think about RX/IF-LO frequency setup etc: Old-style way have been via *setup files created in the /share/obs4/usr/vlbi area with appropriate IF-setup commands. These can be “sourced” directly from a vw% prompt or via CIMA (sending the “source filename” command to the executive) at the time of the observation. **---[TG: update as and when a different way of frequency setting is worked out]**

(E) SNP/PRC file preparation for observation: On vlbis1, run vex2snap in the following way:

1) ./vex2snap /usr2/sched/[FILENAME].vex --station Ar --recorder [RECORDERTYPE]:mk6 --force

where [*FILENAME]* is the HSA/EVN shorthand (usually something like bm440a, or fr059b) and [*RECORDERTYPE]* is *flexbuff* for EVN and *Mark6* for HSA observations, respectively. This should produce a .snap and a .prc program that have the following format: [*FILENAME].*ar (for Arecibo)

(If vex2sanp is not found, then run `source .profile’ in oper@vlbis1 and try again. For further options on vex2snap, use vex2snap --help or talk to Harro)

*******need to update: any other settings need to be regularly run or changed? is --force necessary? how will this change for different types of observations? depends on where the diskpacks are located/make that match*********

2) Move the .prc created to /usr2/proc/
3) A google doc page is available for creating a log of the observations. The link to the log page is
https://docs.google.com/document/d/1qxgFS1VS2MhRSggSFOUd8H4SmHXE5sfqSphgEfdQ1Vc/edit?usp=sharing
Please save the log file as a PDF and reuse the google doc

*******need to update: make a google doc or some other sort of document with a blank form that is easy to fill out with new info for cover sheets for these observations. Should include: file name, date of observation, observer, set up parameters like which receiver, which personality, and which frequency/IF mixer values used (maybe), a place for notes (weather, any difficulties, system problems like a power outage that might change things), the attenuation or power values necessary for calculating Tsys (?), Tsys calculations (?) , and a checkbox indicating that transfer of the resultant files occurred. **********

********note: diskpacks need to be mounted and unmounted in order to be communicated with. add this info********
4) physically place and cable diskpacks, and run mounting commands/OR take steps to verify that the diskpacks are already mounted/set up/ready to go

At the time of running the Observation : [need at least 15 mins of lead time]

Option 0: If you need to run the separate (parallel) continuum data recording on the **Radar Interface (RI)** for continuous Tsys measurement for ampcal, then follow separate notes to cable that system up, and start the data taking there using Phil's "riraw" software.

RI connection details

1. Cable Set-up

FROM	TO
R3,4-11,IN	R4,4-3,IF OUTPUT
R3,4-11,OUT	R3,3-5 (spectrum analyzer),INPUT (50 ohm) Analyzer settings: centre 1.5 GHz, Span 2 Ghz
R4,AMPS 3&4, CH4/ALFA6	R7,7-18,40MHz, left IN
R6,AMPS 3&4, CH4/ALFA6	R7,7-18,40MHz, Right IN
R7,7-18,40MHz, left OUT	R7,7-17, INPUT
R7,7-18,40MHz, right OUT	R8,8-16, INPUT
R7,7-17, either 1,2 or 3	R8,8-12, Square Law Detector 1 IN
R8,8-16, either 1,2 or 3	R8,8-12, Square Law Detector 2 IN DC offset = OFF (middle position) TC = 200 ms
R8,8-12, Square-Law-Detector 1 OUT	R8,8-14,left INPUT
R8,8-12, Square-Law-Detector 2 OUT	R8,8-14, right INPUT Use T-connector
R9,9-4(2nd scope),INPUT 1 (T conn)	R10,10-9,left I
R9,9-4(2nd scope),INPUT 2 (T conn)	R10,10-9,left Q

Use the knob on 7-17 and 8-16 to set threshold IF values.
For bright source 0.5 if it is off-source, and around 1.5 on source
{can someone explain this in bit detail}

After IF, RI analog set up and power adjusted to 0.5 start data taking in observer2

(follow the steps in the old write up)

On the Oscilloscope (9-4): the setup is currently saved under setup1 in the oscilloscope menu, and labelled “VLBI.” If it is deleted or changed, horizontal axis needs to be 20s and vertical 1V.

If you are using RI interface to record data:

- On the radar computer, open observer2.
- In an observer2: window set up background data by
    - gousr
    - cd vlbi
    - riraw - starts data taking
    - back - requests backup file name
    - pntx101 - enters back up file name
    - inpb - inputs the back up file
    - send - sends the back up file to vx-works
    - q - quits the setup program
- In a vw% window (sets up datataking)
    - gousr
    - cd vlbi
    - disc on - turns on disc for recording
    - qdata (checks if there is anything stored here, should be 0), can also be done before “disc on,” but in the old instructions it was written this way
        - If not 0, “mvdata [file location]” use mvdata junk if you think the data here is useless
    - riraw
        - Wait until it says “yes” (like 2 min max)
    - To view the data as it comes in, Phil has an idl script
    - Idl
        - Program is called sravani.pro, found in guest login for the moment
    - WHEN YOU ARE DONE
    - stop
    - mvdata [projectcode] for example “mvdata ta036d”
        - This will create a data file with a .PROJCODE included in the name, in the /share/olda directory

1) Start CIMA, stable mode, enter vlbi **as project code** and your initials, select a receiver (Need to do this **before** FS or if/lo setup). To setup the IF/LO, open Utility from CIMA main menu, then, in “send to Vx-works”, type in -- ‘source setup-filename’ -- (example -- “source n19c3_cja.setup” -- this file should exist in /share/obs4/usr/vlbi)

2) on another computer, vncviewer vlbis1. the password is 123456

3) IF/LO setup if necessary. Baseband recording set up if necessary. Power cycle rdbes if necessary: *rdbe1_reboot.sh* and *rdbe2_reboot.sh* (each of these should report 'on, off, off,

on' as they cycle). The shell files are in /usr2/oper/bin and can be run as oper@vlbis1. From directory /usr2/oper/bin, for example, execute the shell as ./rdbe1_reboot.sh

4) from vlbis1: ssh oper@mark6. the password is FS)@repo

5) on oper@mark6:
Check whether jive5ab is running

Cmd: pgrep jive or ps -elf | grep jive5ab

Sample output:
4 S root 12468 12467 3 80 0 - 103938 ? 13:37 pts/0 00:02:02 /usr/local/bin/jive5ab-3.0.0-64bit-Release -m 3 -b -6 --allow-root
0 S oper 12469 12467 0 80 0 - 1342 - 13:37 pts/0 00:00:00 tee -a /home/oper/jive5ab.logs/jive5ab-Ar.out.2020y042d13h37m21s

If jive5ab is not running then start jive5ab by typing *StartJ5*. This should (soon) report that it is waiting for commands from fs

6) on vlbis1: start the field system by typing *fs*. this should pop open 3 windows--the most important of these is *Operator Input*, and the other two are an error-reporting log and an overview window that displays upcoming time of commands and whether the telescope is on source

7) Start the schedule: in Operator Input, type *schedule=[FILENAME]ar,#1*

8)
************maybe before starting the schedule. Also, need to verify this works******* Test for a sync error by typing *rdbe_cmd=dbe_dot?* If sync_error is not 0, then *****what do?****

Syntax: (https://www.haystack.mit.edu/tech/vlbi/digital/dbe_memos/012.1.pdf)
rdbe_cmd=<dbe>,<timeout>,<actual RDBE command>

Example: to query dbe0, timeout 1 centisec, command dbe_dot?
rdbe_cmd=dbe0,1,dbe_dot?

8) When the observation is complete, in Operator Input, type *terminate*

9)In FS type proc=ft037ar and then run step01. After that adjust power by running the scripts below.

Run RDBE2_AGC_AR.py and RDBE4_AGC_AR.py to adjust the attenuation values to get RMS values of 20 (we decided to keep it 20). These scripts need to be run from oper@vlbis1.

It may be better to adjust the power while the telescope is waiting to track the source.

10) If you started “riraw”, that needs to be stopped, and move the data using a “mvdata filename” command -- [TG: may not be necessary when 80-Hz cal +ANTAB system starts working]

After observation
1) Data Transfer depending on format
2) Send log files back to the severs, vlbeer or nrao (same areas from where original *vex files were downloaded)
3) test how much space left on disks

APPENDIX — I: Some useful documents

AO VLBI link
https://www.naic.edu/~astro/aovlbi/

Jive5ab command set are available at
http://www.jive.eu/~verkout/evlbi/jive5ab-documentation-1.10.pdf

Jive5ab home page http://www.jive.eu/~verkout/evlbi/

RDBE command set are available at
https://www.haystack.mit.edu/tech/vlbi/digital/dbe_memos/012.1.pdf

Info on RDBE
http://wiki.naic.edu/twiki/bin/view/Main/RdbeBox#LCA

Documentation of vbs_ls http://www.jive.eu/~verkout/flexbuff/README.vbs

Arecibo VLBI manual are available at /home/astro/MANUAL (these need to be update)

APPENDIX — II: Some useful commands

In oper@Mark6:
Df -kh — provides the disk usage information
Vbs_ls -l6 — works like ls in linux on Mk6 file system. It combines all the subfiles and list as a single file (see Appendix I for documentation)

Extracting data

For extracting a portion of the data use the following command in FS (this is useful for fringe test)

mk5=scan_set=:14h55m0.0s:14h55m2.0s
mk5=disk2file=N19C3_ar_No0012.m5a:::w

IMPORTANT: these commands should be issued when Mark 6 is not recording data (i.e in between scans)

The extracted file will be available at oper@mark6.naic.edu:/home/oper/data or in Mark6 dir /home/oper/data

Ftp the fringe test data directly from Mark6 using: ftp fringetest.jive.nl, anonymous, email, then cd ftpdata, put filename

Some useful commands

In Mark 6: mount gives the list of disks that are mounted
Then use ls to see the dir corresponding to the scan name is created. Files within that dir should be increasing in size.

From Uwe: How to check RDBE quantization.

dbe_ddc_quantize?0 dbe_ddc_quantize?1 dbe_ddc_quantize?2 ... and so on check the Sampler thresholds on the individual BBCs. The result looks like this:
2019.287.11:-828:1.02/rdbe_cmd/!dbe_ddc_quantize?0:0:42762955:84783105:85232737:43221202:943:-914:14
2019.287.11:59:51.04/rdbe_cmd/!dbe_ddc_quantize?0:1:42815884:84843075:85100373:43240667:1017:-1022:-2
2019.287.11:59:51.04/rdbe_cmd/!dbe_ddc_quantize?0:2:42862198:84687903:85119833:43330065:820:59:5-4
2019.287.11:59:51.05/rdbe_cmd/!dbe_ddc_quantize?0:3:43041103:84581830:84982830:43394236:491:-491:0
Here you see the numbers 42815884:84843075:85100373:43240667 which should have roughly the ratios like the bits 18 32 32 18 or so

To set threshold for quantization (0 here is the first DDC; 1 for second DDC and so on)
dbe_ddc_quantize=0:+thhold:zero:-thhold

The thhold values are 16 bit signed values ranging from -32768 to 32767.

Run the program /usr2/oper/bin/ ddc_quantize_adj.py to get values for threshold. You will need to first adjust the power level using RDBEx_AGC_AR.py (x=2,4) and then include the output of the dbe_ddc_quantize? in the python program to get the correct threshold.

Antenna control programs are in oper@vlbis1:/usr2/st-0.0.0/antcn

Command to list mk6 file in oper@Mark6
Vbs_ls -l6
Eg. vbs_ls -l6 gs* gives the files starting with scan name ‘gs’

To update vex2snap version run `git pull’ in /usr2/RDBE/ in oper@vlbis1

oper@Mark6:bin/vdif_check_data to check quantization in vdif file
Usage:
./vdif_check_data /mnt/disks/4/1/data/gs044b_ar_no0274/gs044b_ar_no0274.00000009 -n 3
Output: (this output is from a file before fixing the quantization issue)

2019y302d22h55m07.000s ,frame_nr = 316, thread_id = 2, nchan = 1, invalid = 0, legacy = 0, station = , bps-1 = 1, data_size = 5032
Thread 0, channel 0, sampler stats: 0 : 28680585(49.7%) 1 : 0(0%) 2 : 145275(0.252%) 3 : 28834140(50%)
Thread 1, channel 0, sampler stats: 0 : 28899999(50.1%) 1 : 0(0%) 2 : 150704(0.261%) 3 : 28629297(49.6%)
Thread 2, channel 0, sampler stats: 0 : 28690304(49.7%) 1 : 0(0%) 2 : 145109(0.252%) 3 : 28844587(50%)

Appendix : Some useful RDBE commands and some useful info on RDBE configuration

The correct syntax of the rdbe_cmd command is this:
rdbe_cmd=<dbe>,<timeout>,<actual RDBE command>

You must always tell the FS *which* rdbe you want to send the command to ("<dbe>" - i.e. "dbe0" or "dbe1") and what timeout (in centiseconds) to apply before giving up waiting for a reply ("<timeout>").

So this would be valid:

rdbe_cmd=dbe0,1,dbe_dot?

RDBE configuration files

These are located at oper@vsibs2:/rfs/rdbe_image/home/roach/personalities/

Note: copied ddc_1601583_nov13_2019_from_Hichem.conf as ddc_1601583.conf on March 10, 2020.

ADC samples, histogram, input bandshape and rms

RDBE command to get ADC samples:
dbe_raw_capture=; or
dbe_raw_capture=[IF(optional)]:[filename(optional)];

This will write 8K ADC samples at the location
oper@vlbis2:/rfs/rdbe_image/home/roach/rdbe_sw/data_capture/raw_capture_date_time.txt or with the provided filename. IF=0 or 1; if not specified it assumes 0.The RDBE filesystem is mounted in vlbis2 and so you need to login to vlbis2 with naic305

RMS of the ADC samples are given by
Dbe_rms?
!dbe_rms=0:4.002:2.089:-0.272:-1.381:15:16;
=0:Pol1rms:Pol2rms:xx:xx:xx:xx;

Appendix: VLBISH

"vlbish" program can communicate with the RDBEs. "vlbish" program is installed on the FieldSystem computer. Type
$> vlbish
Welcome to vlbish Id [history nomysql noevlbilookup]
q to quit, h for help
>rdbe2/dbe_rms?
rdbe2>
Or
>rdbe2,rdbe4/dbe_rms?
rdbe2,rdbe4>

Here you can enter RDBE commands.

you can address your RDBEs directly like this:
> fs rdbe 0/dbe_dot?

In words this reads: "send the dbe_dot? command to the RDBE that is configured as "rdbe0" in the FieldSystem". The "vlbish" program will actually parse "/usr2/fs/control/rdbead.ctl" to resolve rdbe number 0 and rdbe 1.

Issues with recording data to Mk6

Data recording in Mk6 can be checked by typing
Vbs_ls -l6 | grep filename - filename is same as the prefix name of the vex or snap file

If data is not recording then check whether rdbes are sending data. In vlbish type the command

Rdbe2, rdbe4/dbe_data_send?

Output - on:YYYYDOYDDHHMM:YYYYDOYDDHHMM:YYYDOYDDHHMM:;

The first `on' indicates the data is being sent, the first date indicates the start date of the data being sent, the second data indicates the stop date for the data to be sent and the last date is the current date and time.

If the start date is in future then RDBEs will not send data. One can use dbe_data_send to set the start, stop dates. See the help for dbe_commands in the online document or check the log file created under /usr2/log/filename.log (filename is the prefix of the snap file).

Mattermost for EVN chat

https://coms.evlbi.org/login?redirect_to=%2Fevn%2Fpl%2F3nxibfhuktr6p83f6ckjhzj9ue aroshi@naic.edu pwd

Arecibo Clock information

http://www.naic.edu/aoclock (This file is maintained by Felix Fernandez)

FieldSystem UT time synchronization

Use this command from the "operators" input window of FS

sy=run setcl computer &

(the "spacings" above are very important).

A lot of information on FS's own time keeping can be found on the computer where fs is run, in this file: /usr2/fs/misc/fstime.txt

Transfering the VLBI data

For EVN, Martin Leeuwinga (leeuwinga@jive.eu) will e-ship the data from Arecibo Mark6 disks. The calibration data need to be sent separately to Martin. Some useful communication on e-shipping is given below.

For HSA:

It can be transferred via ftp using the following instructions (The files are at The files are at oper@mark6:/home/oper/data):
ftp ftp.aoc.nrao.edu

Name: anonymous
331 Anonymous login ok, send your complete email address as your password
Password: (for example, mclausse@nrao.edu)

ftp> cd incoming/mclaussen/arecibo

250 CWD command successful

ftp>bin

200 Type set to I

ftp>put <filename>

221 Goodbye

For HSA, we have to ship the disk.
----------Communication of e-shipping----------
Hi Arun, Luis,

I've just tried a manual e-shipping of what we would do automatically.
I logged in to one our flexbuffs at JIVE (flexbuf4) and transferred one of the DDC8 recordings I just made:

[flexbuf4]Okay-> m5copy -r 100M -udt vbs://192.231.96.196/mk6/ddc8_hv_no0003 vbs://192.42.120.39/

and left it running (I went on to do other things).
The "-r 100M" tells the software to transfer at at most 100Mbps, never faster. If reality is less, well that is a pity but no harm done.

Then I came back to the terminal and it had finished. Using the "vbs_ls" program on flexbuf4:

```
[flexbuf4]Okay-> vbs_ls -lh ddc8_hv*
Found 1 recordings in 20 chunks,    5.00G
drw-r--r-- jops      flexbuf      5.00G Jun 27 19:38 ddc8_hv_no0003
```

Doing the same on mark6.naic.edu:

```
oper@Mark6-4XXX:~$ vbs_ls -6lh ddc8*
Found 4 recordings in 48 chunks, 12.00G
drw-r--r-- oper oper 2.00G Jun 27 18:22 ddc8_hv_no0001
drw-r--r-- oper oper 3.50G Jun 27 18:34 ddc8_hv_no0002
drw-r--r-- oper oper 5.00G Jun 27 18:45 ddc8_hv_no0003
drw-r--r-- oper oper 1.50G Jun 27 19:20 ddc8_hv_no0004
```

So ... welcome to the e-shipping EVN :-)

Email from Uwe (steps to be done after observing)

We have a list of so called permanent action items for VLBI friends that list the things that should be done before, during and after the session:
https://deki.mpifr-bonn.mpg.de/Working_Groups/EVN_TOG/Permanent_Action_Items

And this includes the upload of log files to the ftp-server
vlbeer.ira.inaf.it
As well as the antabfs files for calibration.
Do you know the account and password?

The FS provides a script to send the log after the observation with the sched_end procedure. The script is called lgput and is available in

/usr2/fs/misc/lgput

A description of how to use it is in
/usr2/fs/misc/lgput/simple_logsend.txt

Best wishes,
Uwe

Cheers!
H

Anish, Martin:

Peak bandwidth out of AO is 600Mbps. 300Mbps should be OK (and I have seen sustained 30MBytes/s transfers over rsync/TCP from sites).

As you can imagine the internet connections (probably all over the known universe) are pretty busy these days; the network is more than the telescope, it's a way of life:(

Regards,
-arun

----------Mail from Uwe & Harro on how to check Mark6 data (Uwe's mail Feb 13, 2020)

Hi all,

Since the beginning of the week I've been able to install the DiFX utilities "mark5access" and "vdifio". Don't ask how.

Nothing works out of the box. And now that e.g. "m5spec" and "m5spec.py" (they're very different!) and "vdif_spec" are installed (/usr/local/bin/) and actually run (*sigh*) they still don't work out of the box.

The only program that might work out of the box is "m5spec" but all that that produces is a "csv" text file with the amplitudes for each spectral point for the channels. It doesn't plot them.
"m5spec.py" has subtly different command line options but does use matplotlib to actually draw the spectra on screen.

Despite their name(s), "m5spec*" can handle MarkIV/VLBA, Mark5B and _some_ VDIF data. Neither of these can deal with the multi-thread VDIF coming out of the RDBE's DDC mode. That's where "vdifspec" could come in: it runs another DiFX utility "vmux" (VDIF multiplexer) which you can tell to turn the 8 threads / 1channel per thread VDIF into 1 thead with 8 channels in it and then pipe it into "m5spec" (because the only flavour of VDIF that one accepts is single-thread VDIF ...), after which that program dissects the multi-thread VDIF back into single channels *sigh*.

Anyway, I'd be fine with that, if it would work. Only it doesn't. It segfaults.
If I do the two steps manually, "vmux" into single thread VDIF into a file on disk and then call "m5spec" on that created file it does NOT segfault but produce something.
Unfortunately there is no (command line) control over how much data to process so unless we first do "disk2file" to extract a smaller chunk of data this won't be able to work in production.

And finally, looking at the plots generated from the "vmux"ed data (it should have 8x32MHz channels - DDC8 was used) and the (attached) plot does have that but they look awful, more like 16MHz bands. Not really happy about that.

I also attached an example of the pfb output plot (pfb.png) - it is very empty but that's because probably the data that came out of the RDBE/PFB at that time was pretty zero.

The ddc8.png was produced as follows:

```
        # convert 8thread x 1chan into 1thread x 8chan manually
        oper@Mark6-4XXX:~$ vmux /tmp/data/test_hv_no0020 5032 3200 0,1,2,3,4,5,6,7
/tmp/test_hv_no0020.vmux
        # use m5spec.py to plot to screen, then use "save as" button
        oper@Mark6-4XXX:~$ /usr/local/bin/m5spec.py /tmp/test_hv_no0020.vmux
VDIF_40000-1024-8-2 32 1024
```

The pfb.png was generated like this:

```
        # PFB outputs Mark5B format so can take it directly from "file under vbs_fs
mountpoint"
        oper@Mark6-4XXX:~$ mkdir /tmp/data
```

oper@Mark6-4XXX:~$ vbs_fs -6 /tmp/data
oper@Mark6-4XXX:~$ /usr/local/bin/m5spec.py /tmp/data/fr059c_ar_no0059 mark5b-2048-16-2 23 1024
oper@Mark6-4XXX:~$ fusermount -u /tmp/data

Cheers,
h

Hi Anish,

I don't know if you already have this tools and know all this, but just to give you some ideas how to look at data that you recorded locally I've prepared here some description. To analyze data I use some tools from the DiFX correlator package. There is vmux in the vdifio library that converts multi-thread VDIF to single thread and the mark5access library that provides some programs to analyze single thread VDIF. The SVN repository is available here

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

The whole package has about 3 GB, but you could also just download

https://svn.atnf.csiro.au/difx/libraries/codifio/
https://svn.atnf.csiro.au/difx/libraries/mark5access/
https://svn.atnf.csiro.au/difx/libraries/vdifio/

I could not compile the mark5access without the codifio, so I put it here as well. The libraries also require some packages like FFTW3 and also some that provide tools to produce the configure script (e.g. libtools, automake). I'm not very familiar with such things, but just followed the instructions in the README, like

```
aclocal -I m4
libtoolize --copy --force
autoconf
autoheader
automake -a -c

./configure --enable-python --prefix=${DIFXROOT}

make
# (su root?)
make install
```

The vdifio also provides some utilities like vdifbstate or vdifspec that can be used on the multi-threaded VDIF data directly. It is a scripts that calls vmux and the corresponding mark5access tool ,e.g, for a single RDBE 4x128 MHz data

```
oper@Mark6-4040:~$ vdifbstate test_eb_no0001.m5a 5032 2048 0,1,2,3 500
Executing: vmux test_eb_no0001.m5a 5032 12800 0,1,2,3 - 0 | m5bstate - VDIF_20000-2048-4-2 500
```

```
Mark5 stream: 0x1ddb140
  stream = File-1/1=<stdin>
  format = VDIF_20000-2048-4-2 = 3
  start mjd/sec = 58904 47833.101406250
  frame duration = 78125.00 ns
  framenum = 0
  sample rate = 256000000 Hz
  offset = 0
  framebytes = 20032 bytes
  datasize = 20000 bytes
  sample granularity = 1
  frame granularity = 1
  gframens = 78125
  payload offset = 32
  read position = 0
  data window size = 524288 bytes
10000000 / 10000000 samples unpacked

Ch   --      -     +     ++       --      -     +     ++     gfact
 0 1696545 3304452 3309563 1689440     17.0   33.0   33.1   16.9   1.05
 1 1700972 3301849 3306131 1691048     17.0   33.0   33.1   16.9   1.05
 2 1695685 3300442 3308420 1695453     17.0   33.0   33.1   17.0   1.05
 3 1693581 3299236 3310650 1696533     16.9   33.0   33.1   17.0   1.05
```

Attached is also a vex-file that you can use to produce a prc-file with setups for different bandwidth, ranging from 8 to 64 MHz BBCs, 256-2048 Mbps recording rates. It is not meant as a schedule to run, to record data, but you can use the prc-file to configure the RDBEs and record some data by hand. I hope I didn't make any errors in the schedule. e.g. to use it

```
proc=rdbe01ar
setup04
# check that everything is configured fine. IF level, BBC counts...

mk5=record=on:test_ar_no0001
mk5=record=off
mk5=scan_check?
mk5=scan_set=::+1s
mk5=disk2file=:::w
```

This will write 1s of data to your local disk, test_ar_no0001.m5a, and you can play with it. Of course you don't have to write out the data to the local disk and can work on the disk pack directly. Just take one of the files on /mnt/disks/1/0/data or where ever it is mounted. I usually use the way above, because the path-name is shorter when I work in a local directory.

Cheers,
Uwe

----------Mail from Uwe on the RDBE tuning----------

The RDBEs can output 32 MHz DDC channels and your prc-file looks
correct. At least for the bandwidth

rdbe_dc_cfg=dbe0,0:4:21.00:0;
rdbe_dc_cfg=dbe1,0:4:21.00:0;
rdbe_dc_cfg=dbe0,1:4:11.00:1;
rdbe_dc_cfg=dbe1,1:4:11.00:1;
rdbe_dc_cfg=dbe0,2:4:43.00:1;
rdbe_dc_cfg=dbe1,2:4:43.00:1;
rdbe_dc_cfg=dbe0,3:4:75.00:1;
rdbe_dc_cfg=dbe1,3:4:75.00:1;

#DBE #BBC #dec #freq #basebandmode
dbe0,0:4:21.00:0;

The #dec parameter is the decimation rate.

1 corresponds to 128 MHz filters
2 corresponds to 64 MHz filters
4 corresponds to 32 MHz filters
8 corresponds to 16 MHz filters
...

I'm a bit surprised that the synthesizer freqs are integers. The
vex-file asked for frequency channels with 0.5 MHz at the end, e.g. 1594.50

2400-1594.50= 805.5
This will fall in the band 768-896 MHz and should produce
rdbe_dc_cfg=dbe0,0:4:21.50:0;

The next one should be rdbe_dc_cfg=dbe0,1:4:10.50:1;

Maybe a problem with some float to integer conversion?

> 1. Do our RDBEs support 32 MHz per DDC channel?
> 2. Assuming RDBEs support 32 MHz bandwidth -- we are then using only one
> RDBE for this experiment. Is this correct?

Each RDBE outputs only 4 channels, so you need one RDBE per polarization.

> 3. Why are there LSB and USB in the correlator output? Will the
> correlator folds the 32 MHz and unfolds into LSB and USB?
>
yes, the correlator "zooms" your wider bands to the 16 MHz DBBC bands.
So the first half of the 32 MHz USB gets a 16 MHz LSB band and the
second half is just correlated with the 16 MHz USB of the DBBC.

Cheers,
Uwe

Not really. You have two inputs and both IFs get split into sub-bands from 512-640, 640-896, 896-1024. The central one has 256 MHz of bandwidth and allows a bit more flexibility were to set the DDC channels as long as they are narrower than 128 MHz.

The dbe_xbar command decides which IF and part of the band goes to which DDC channel. e.g. from your prc-file all channels from each RDBE are connected to one IF and one part of the band:
rdbe_cmd=dbe1,3,dbe_xbar=7:7:7:7:2:2:2:2;
rdbe_cmd=dbe0,3,dbe_xbar=3:3:3:3:2:2:2:2;

The counting is arbitrarily complicated:

<xbar_position>:<IF>:<freq. range>
2:0:512-640
1:0:640-896
3:0:896-640 (#1, reversed frequency ordering)
0:0:896-1024
6:1:512-640
5:1:640-896
7:1:896-640 (#5, reversed frequency ordering)
4:1:896-1024

So you could run a 4 DDC mode with one RDBE and have two BBCs on IF 0 and the other on IF 1.

To produce 8 BBCs you have to use two RDBEs. They can be on the same polarization or on two different ones. And with 8x128 MHz you get 4 Gbps. This is the maximum the VLBA can do. The current 8x32 MHz produces 1024 Mbps.

Cheers,
Uwe

_Debugging the system: How to acquire a short data with RDBE and Mk6 recorder?

Rdbe01.vex is a file provided by Uwe which has 8,16,32 MHz setups for testing. One can use vex2snap with '--shift today' option which will shift the date of obs to today.

vex2snap /usr2/sched/[FILENAME].vex --station Ar --recorder [RECORDERTYPE]:mk6 --force --shift today

Sometimes the 'shifting to today' won't work since the test time slot may not match the scan time in the vex file.

24. Create snp and prc file (for EVN use flexbuff; for HSA use Mark6)
25. Check enough disk space is available in Mark6 (In oper@mark6 df -kh). Delete old files to create space if there is a shortage. For fringe test, remove old files from /home/oper/data (cmd rm *)
26. Start cima, user vlbi, your initial, line; select receiver; run setup
27. Vncviewer vlbis1:2 (123456)
28. In vlbis1: ssh oper@mark6 (naic305m); check jive5ab is running (ps -elf | grep jive)
29. Reboot RDBEs (/usr2/oper/bin/*rdbe2_reboot.sh* and *rdbe4_reboot.sh*)
30. **Wait till RDBEs reboot** (Check they have rebooted by issuing, for example, dbe_rms? From vlibish)
31. Run FS (from /usr2/oper); check FS time is synched with UT time (if not follow the steps to synch it) In FS, proc=xxxar (xxar.prc is the prc file)
32. In FS, setupxx (eg setup01 for setup01 in the prc filpe). The setup01 file is usually located in /share/obs4/usr/vlbi/XXX.setup
33. **Wait till RDBEs are initialized**
34. In vlbis1: run /usr2/oper RDBE2_AGC_AR.py and RDBE4_AGC_AR.py to set RMS to 20
35. In vlbis1: vlbish - run rdbe2, rdbe4/dbe_rms? To get the RMS values
36. In vlbis1: vlbish - run command rdbe2,rdbe4/dbe_ddc_quantize?
37. In vlbis1: run /usr2/oper/bin/ddc_quantize_adj.py to get threshold values
38. In vlbis1: vlbish - run command dbe_ddc_quantize=x:+th:zth,-th, for x=0-3
39. Check for sync err (vlbish dbe_dot?); if not synced in vlbish dbe_dot_set=;
40. Check RDBEs are sending data. Dbe_data_send?
41. If data is not send issue dbe_data_send=on:yyyydddhhmmss:yyyydddhhmmss where yyyy is year, ddd is day number, hhmmss are UT hours, minutes and second from start to end of data sending.
42. After dbe_data_send? Gives `on' status, In FS issue the following commands
    mk5=record=on:test_ar_no0001
    mk5=record=off
    mk5=scan_check?
    mk5=scan_set=::+1s
    mk5=disk2file=:::w
43. In oper@mark6 vbs_ls -l6 | grep test_ar* to check data is being written when record=on command is issued.
44. The above commands will create a MK6 file in /mnt/…. And a file test_ar_no0001.mk5a in /home/oper/data/
45. Analyze the data using vmux and mk5spec.py programs

RDBE & FILE SYSTEM AT AO

RECORDER MARK6 AT AO

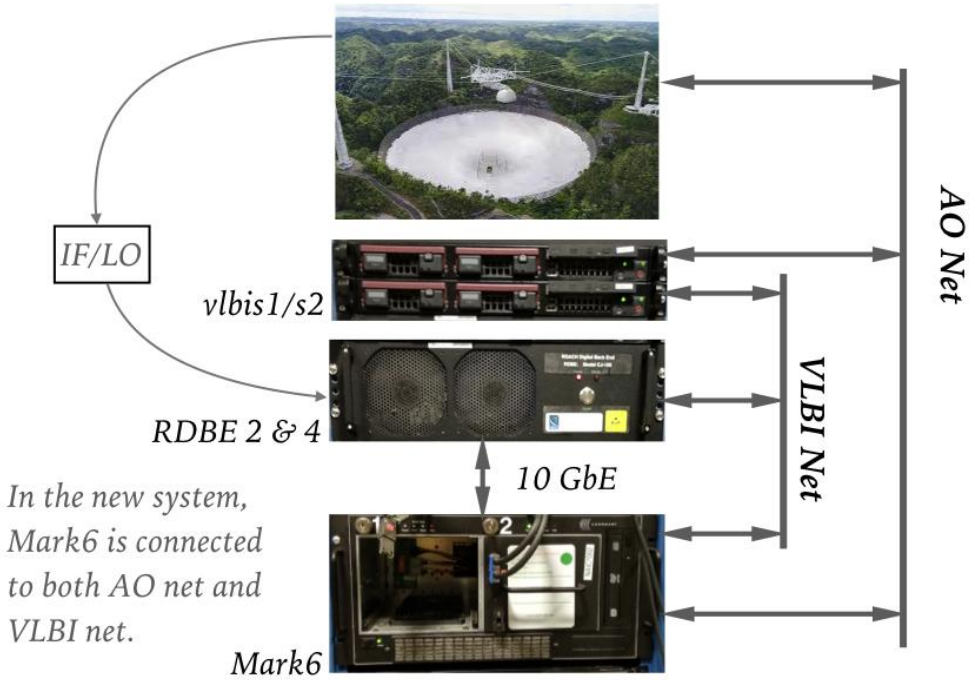
Mark6 extension

Mark6

*Mark6 has two disk controllers. Each of them carries two pairs of cables (labelled 1,2,3,4 in the picture). Each pair drives eight disks (disk pack). Each of the eight disks is of 1 Tb. The label on the disk pack will have a * indicating that it is for Mark6.*

Always unmount the disk pack before turning the power key off.

RDBE AND MARK6 DATA COMMUNICATION



In the new system, Mark6 is connected to both AO net and VLBI net.

RDBE & MARK6 NETWORK

The 10 GbE switch has been removed between RDBE and Mark6 and both the RDBEs are connected directly to Mark6 using 10 GbE