Python control of the DBBC3 backend

EVN TOG Meeting 2023, Bonn

Helge Rottmann, MPIfR

Max-Planck-Institut für Radioastronomie

MAX PLANCK GESELLSCHAFT

The DBBC3 backend can be controlled and monitored from python via the `dbbc3` package

The package is available on github: https://github.com/mpifr-vlbi/dbbc3

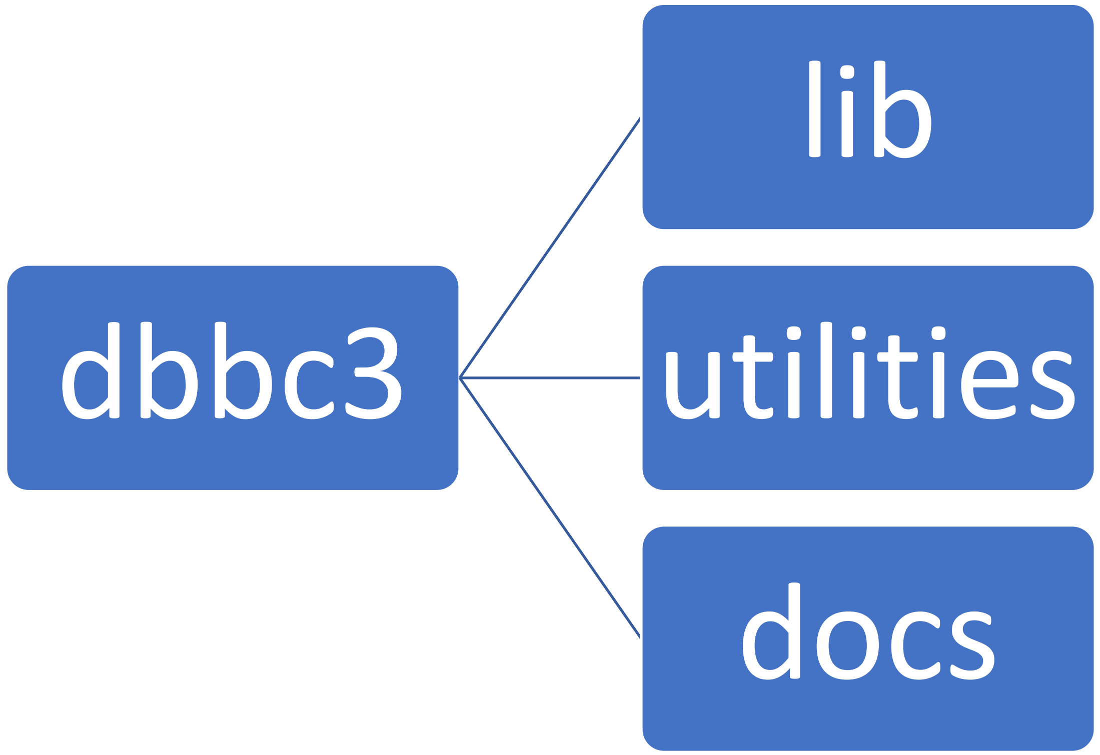
Current stable version: 0.3

Installation:

```
git clone https://github.com/mpifr-vlbi/dbbc3.git
cd dbbc3/lib
python setup.py install
```

Documentation is available here: https://dbbc3.readthedocs.io

dbbc3 package structure



the dbbc3 site-package
(will be installed system-wide by
`python setup.py install`)

A collection of various standalone tools and utilities

Source files for generating documentation with sphinx

Using the dbbc3 site-package

Simple example obtains IF settings of board A (=0)

```
from dbbc3.DBBC3 import DBBC3

dbbc3 = DBBC3("134.104.30.223")
dbbc3.dbbcif(0)
```

Output:

```
Selecting commandset version: DBBC3Commandset_DDC_U_126

{'inputType': 2, 'attenuation': 4, 'mode': 'agc', 'count': 31381, 'target': 32000}
```

Note:
command set is attached dynamically matching the currently running DBBC3 control software

dbbc3 command logic

The package tries to replicate the „native“ DBBC3 command names as closely as possible:

dbbc3 native command	Python equivalent
time	dbbc3.time()
dbbcifa=2,10	dbbc3.dbbcif(‘A‘, 2, mode=10)
core3h=1,sysstat	dbbc3.core3h_sysstat(0)
adb3l=reseth	dbbc3.adb3l_reseth()

Note:

- For python commands *board* and *sampler* numbering ***always starts at 0*** (for native commands boards start at 1)
- Boards can be specified either by integer numbers or characters: 0=A, 1=B etc.

dbbc3 validation

The dbbc3 package provides higher level validation methods via the DBBC3Validation module e.g. for:

- Checking IF settings
- Checking sampler settings
- Checking synthesizer settings

see documentation for a the full list of validation methods

```
from dbbc3.DBBC3 import DBBC3
from dbbc3.DBBC3Validation import ValidationFactory

dbbc3 = DBBC3("134.104.30.223")
valFactory = ValidationFactory()
val = valFactory.create(dbbc3, True)

val.validateSynthesizerLock(0)
```

dbbc3 validation cont.

Validation methods return a ValidationReport object

The ValidationReport can contain multiple Item entries

```
Item properties
action:        a description of what was validated
state:         the outcome of the the validation
level:         the logging level of the validation
message:       the validation outcome message
exit:          True if the validation should trigger an exit event
resolution:    A message describing possible solutions for failed validations
```

```
...
rep = val.validateSynthesizerLock(0)
print(rep)
```

```
action:     === Checking synthesizer lock state of board A
state:      OK
level:      INFO
message:    Locked
exit:       False
resolution:
```

dbbc3 multicast

The DBBC3 sends multicast messages containing its current state on a one second cadence.

Supported software versions:

DSC versions >= 120

DDC versions >= 125

OCT versions >= 120

The content of the multicast message is mode-dependent

dbbc3 multicast processing

The dbbc3 package provides the `DBBC3Multicast` module which handles processing of multicast messages

```
from dbbc3.DBBC3Multicast import DBBC3MulticastFactory

mcFactory = DBBC3MulticastFactory()
mc = mcFactory.create()

message = mc.poll()
```

The multicast message is returned as a dictionary

dbbc3 multicast message

Example multicast message dict (OCT_D mode)

```
{'mode': 'OCT_D’, 'majorVersion': 120, 'minorVersionString': 'August 31st 2022',
'minorVersion': 220831, 'boardPresent': [True, True, True, True, False, False, False,
False], 'boardActive': [True, True, True, True, False, False, False, False], 'if_1':
{'mode': 'agc', 'attenuation': 11, 'count': 31882, 'target': 32000, 'synth': {'status':
1, 'lock': 1, 'attenuation': 18, 'frequency': 4524.0}, 'sampler0': {'power': 72746343,
'offset': 64410282}, 'sampler1': {'power': 73962686, 'offset': 63665610}, 'sampler2':
{'power': 73158462, 'offset': 63718535}, 'sampler3': {'power': 73743109, 'offset':
63949517}, 'delayCorr': (147462423, 144809580, 148870960), 'vdiftime': 2058959,
'vdifepoch': 46, 'ppsdelay': 999999984, 'filter1': {'power': 121762240, 'stats':
(22683397, 41360632, 41210614, 22745357), 'statsFrac': (17.72140390625,
32.312993750000004, 32.1957921875, 17.76981015625)}, 'filter2': {'power': 166743416,
'stats': (21283596, 41831470, 40420110, 24464824), 'statsFrac': (16.627809375,
32.6808359375, 31.5782109375, 19.11314375)}},
```

…

dbbc3 utilities

utility	purpose
dbbc3client.py	An interactive client for communicating with the DBBC3
dbbc3ctl.py	A general purpose tool to validate the state of the DBBC3 system or its sub-systems
dbbc3mon.py	A GUI tool for monitoring the DBBC3 (requires multicast)
dbbc3_powerlogger.py	Logs the DBBC3 power readings (requires multicast)
convert_powerlog_to_HDF5.py	Convert the DBBC3 power log files to HDF5 format
dbbc3_ppslogger.py	Logs the DBBC3 PPS delays (requires multicast)

dbbc3 utilities – dbbc3ctl.py

dbbc3ctl.py: validate the state of the DBBC3 system and/or its sub-systems

Interactive mode

```
> ./dbbc3ctl.py 134.104.30.223
=== Trying to connect to 134.104.30.223:4000
Selecting commandset version: DBBC3Commandset_DDC_U_126
=== Connected
=== DBBC3 is running: mode=DDC_U version=126(221103)
=== Using boards: [0, 1]
Welcome to the DBBC3.  Type help or ? to list commands
(dbbc3ctl): ?
check recorder @host @interface
check sampler offset [all,0,1]
check sampler gain [all,0,1]
check sampler phase [all,0,1]
check timesync [all,0,1]
check synthesizer lock [all,0,1]
check synthesizer freq [all,0,1]
check bstate [all,0,1]
check pps
check system [all,0,1]
get version
```

dbbc3 utilities – dbbc3ctl.py

Full system validation:

```
> ./dbbc3ctl.py 134.104.30.223
=== Trying to connect to 134.104.30.223:4000
Selecting commandset version: DBBC3Commandset_DDC_U_126
=== Connected
=== DBBC3 is running: mode=DDC_U version=126(221103)
=== Using boards: [0, 1]
Welcome to the DBBC3.  Type help or ? to list commands
(dbbc3ctl): check system all
...

[OK] === Checking sampler phases -
=== Checking board 0
[OK] === Checking 1PPS synchronisation < +- 200 ns - PPS delays: [16, 16] ns
[OK] === Checking time synchronisation of core board A - Reported time: 2023-01-24 12:53:38
[OK] === Checking synthesizer lock state of board A - Locked
[OK] === Checking GCoMo synthesizer frequency of board A - Freq=9000.000000 MHz
[WARNING][FAIL]/[WARN] === Checking IF power level on core board A - IF input power is too low.
The attenuation should be in the range 20-40, but is 4
[RESOLUTION] Increase the IF power
...
```

dbbc3 utilities – dbbc3ctl.py

Scripted mode

Execute a single command

```
> ./dbbc3ctl.py -c "check synthesizer lock" 134.104.30.223
```

Execute multiple commands

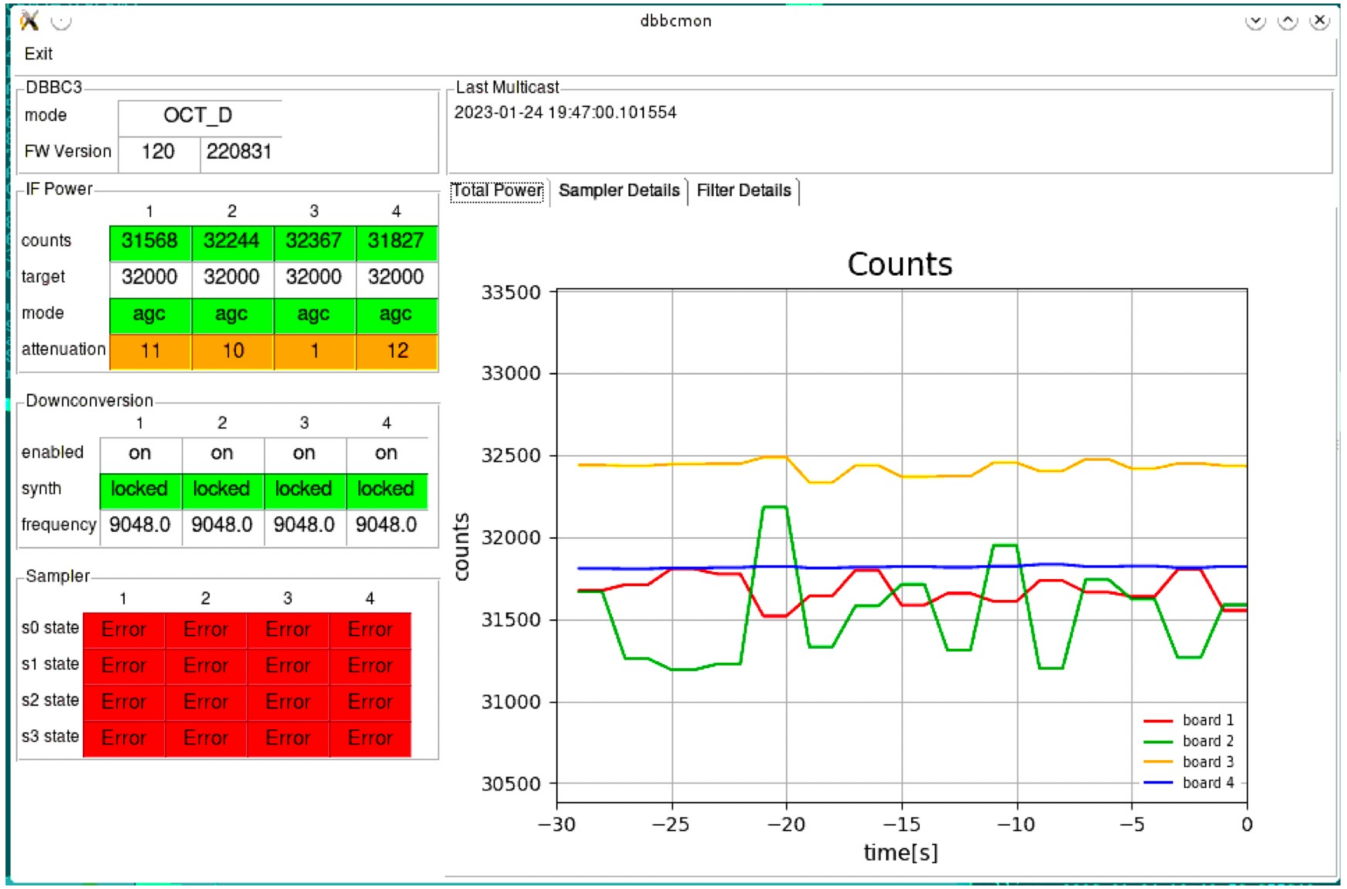
```
> ./dbbc3ctl.py -c "check synthesizer lock" –c "check synthesizer freq" 134.104.30.223
```

Execute command multiple times (e.g. 10)

```
> ./dbbc3ctl.py -c "check synthesizer lock" –r 10 134.104.30.223
```

dbbc3 – utilities dbbc3mon.py

GUI tool for monitoring the DBBC3 state (not fully implemented yet)



DBBC3

mode	OCT_D	
FW Version	120	220831

Last Multicast

2023-01-24 19:47:00.101554

IF Power

	1	2	3	4
counts	31568	32244	32367	31827
target	32000	32000	32000	32000
mode	agc	agc	agc	agc
attenuation	11	10	1	12

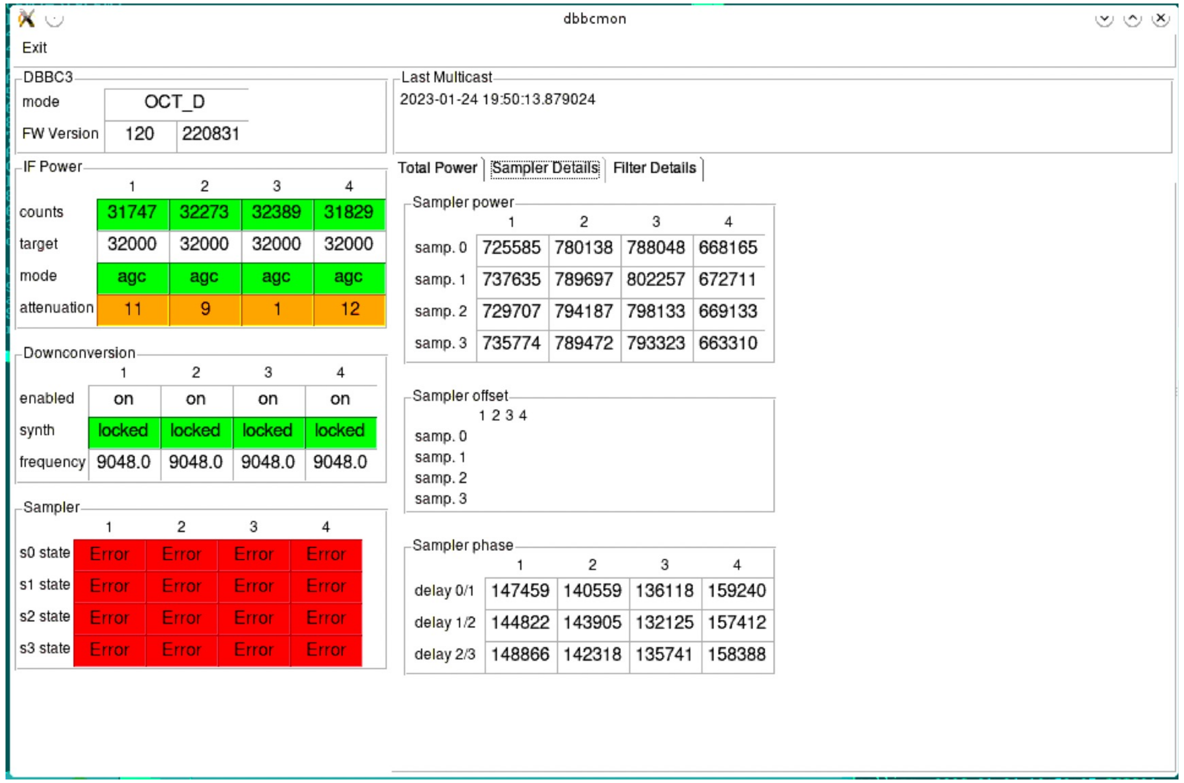
Downconversion

	1	2	3	4
enabled	on	on	on	on
synth	locked	locked	locked	locked
frequency	9048.0	9048.0	9048.0	9048.0

Sampler

	1	2	3	4
s0 state	Error	Error	Error	Error
s1 state	Error	Error	Error	Error
s2 state	Error	Error	Error	Error
s3 state	Error	Error	Error	Error

dbbc3 – utilities dbbc3mon.py



dbbcmon

Exit

DBBC3

mode	OCT_D	
FW Version	120	220831

Last Multicast
2023-01-24 19:50:13.879024

Total Power | Sampler Details | Filter Details

IF Power

	1	2	3	4
counts	31747	32273	32389	31829
target	32000	32000	32000	32000
mode	agc	agc	agc	agc
attenuation	11	9	1	12

Downconversion

	1	2	3	4
enabled	on	on	on	on
synth	locked	locked	locked	locked
frequency	9048.0	9048.0	9048.0	9048.0

Sampler

	1	2	3	4
s0 state	Error	Error	Error	Error
s1 state	Error	Error	Error	Error
s2 state	Error	Error	Error	Error
s3 state	Error	Error	Error	Error

Sampler power

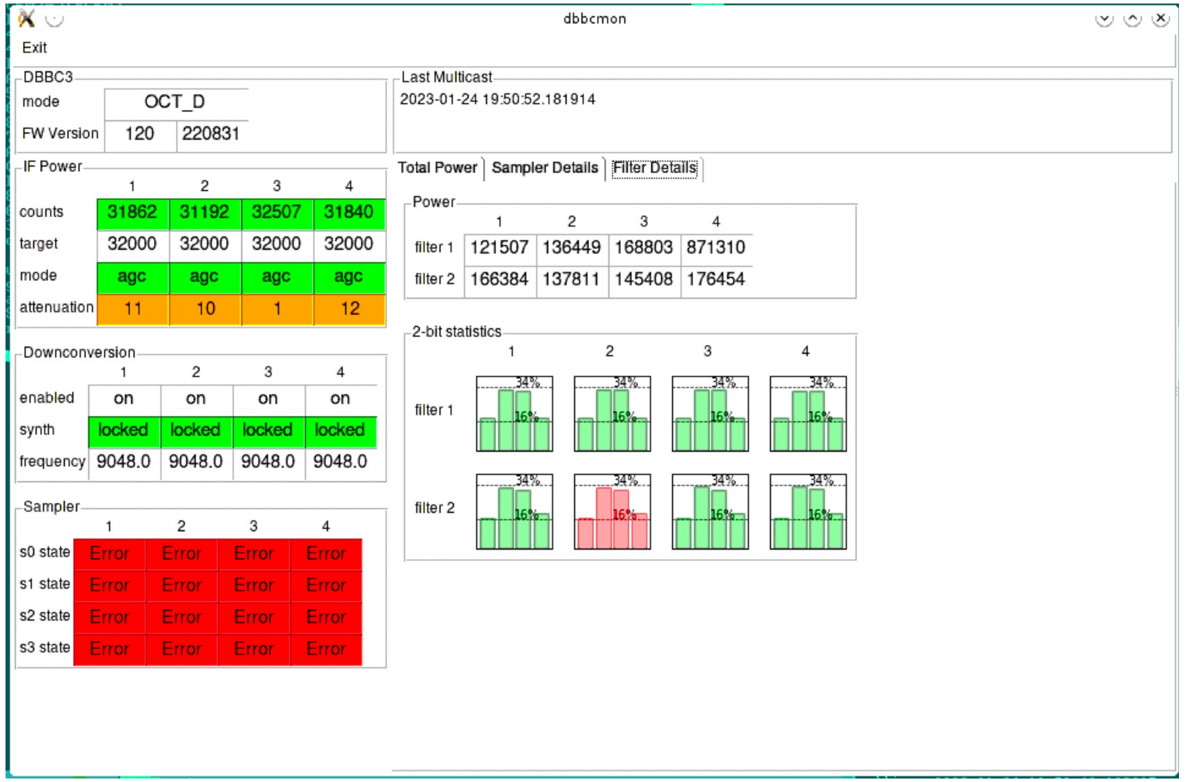
	1	2	3	4
samp. 0	725585	780138	788048	668165
samp. 1	737635	789697	802257	672711
samp. 2	729707	794187	798133	669133
samp. 3	735774	789472	793323	663310

Sampler offset

	1	2	3	4
samp. 0				
samp. 1				
samp. 2				
samp. 3				

Sampler phase

	1	2	3	4
delay 0/1	147459	140559	136118	159240
delay 1/2	144822	143905	132125	157412
delay 2/3	148866	142318	135741	158388



dbbcmon

Exit

DBBC3

mode	OCT_D	
FW Version	120	220831

Last Multicast
2023-01-24 19:50:52.181914

Total Power | Sampler Details | Filter Details

IF Power

	1	2	3	4
counts	31862	31192	32507	31840
target	32000	32000	32000	32000
mode	agc	agc	agc	agc
attenuation	11	10	1	12

Downconversion

	1	2	3	4
enabled	on	on	on	on
synth	locked	locked	locked	locked
frequency	9048.0	9048.0	9048.0	9048.0

Sampler

	1	2	3	4
s0 state	Error	Error	Error	Error
s1 state	Error	Error	Error	Error
s2 state	Error	Error	Error	Error
s3 state	Error	Error	Error	Error

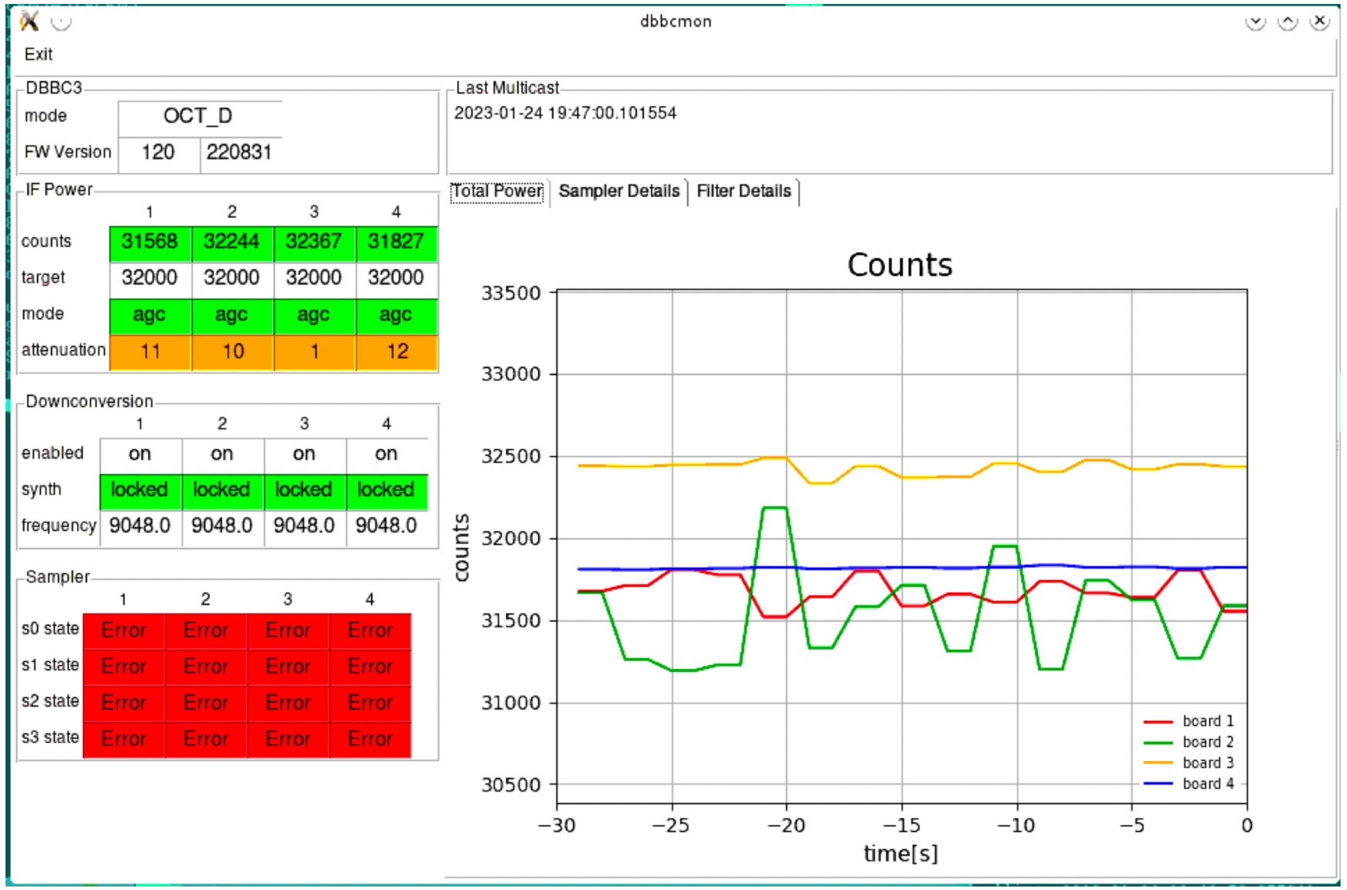
Power

	1	2	3	4
filter 1	121507	136449	168803	871310
filter 2	166384	137811	145408	176454

2-bit statistics

dbbc3 – utilities `dbbc3mon.py`

GUI tool for monitoring the DBBC3 state (not fully implemented yet)



DBBC3

mode	OCT_D	
FW Version	120	220831

Last Multicast

2023-01-24 19:47:00.101554

IF Power

	1	2	3	4
counts	31568	32244	32367	31827
target	32000	32000	32000	32000
mode	agc	agc	agc	agc
attenuation	11	10	1	12

Downconversion

	1	2	3	4
enabled	on	on	on	on
synth	locked	locked	locked	locked
frequency	9048.0	9048.0	9048.0	9048.0

Sampler

	1	2	3	4
s0 state	Error	Error	Error	Error
s1 state	Error	Error	Error	Error
s2 state	Error	Error	Error	Error
s3 state	Error	Error	Error	Error