A level0 to level1b database for Odin-SMR

Joakim Möller, joakim.moller@molflow.com
Bengt Rydberg, bengt.rydberg@molflow.com
Gothenburg, Sweden

May 23, 2012

Contents

1 Overview

The aim of this project has been to make the existing calibration procedure, i.e. the processing of various level0 files to calibrated spectra, of Odin-SMR data more transparent. A database with tables for level0-, level1a-, and level1b data, and data import scripts have been developed (see Fig. 1).

The old calibration code has to a very large extend been re-used, but the code has been partitioned into several smaller scripts, that stores data into database tables. This will not only make further development more simple, the processing can be significant more time efficient as well. For example, if one wants to test a new calibration version (level1a to level1b) one does not have to re-process all data below level1b.

This document describes:
– the database and how to import data into the various database tables
– calibration scheme
– required packages and installations

2 The Database

2.1 Overview

The database consists of nine tables (see Fig. 1):
– four level0 tables (ac_level0, attitude_level0, shk_level0 [housekeeping], and fba_level0 [reference signal type]) that contains data from level0 files,
– three level1 or level1a (attitude_level1, shk_level1, ac_level1a) that contains processed level0 data,
– two level1b tables (ac_level1b that contains calibrated spectrum towards target and ac_cal_level1b that contains sideband ratio spectrum and calibration spectrum)

To clarify the different levels in terms of auto-correlator (ac) data:
– ac_level0 contains the measured correlation coefficients (normalised by integration time)
– ac_level1a contains spectrum in the frequency domain
– ac_level1b contains intensity (and frequency) calibrated spectrum

2.2 Level0 tables

Four level0 tables: ac_level0, attitude_level0, shk_level0, and fba_level0 contains data from level0 files, and the tables are here described in some more details. ac2db, att2db, shk2db, and fba2db are executables responsible for importing data into these tables (how to import data can be seen Sect. 2.5). Satellite time word (stw) is a key-column in all these tables. However, data in the different level0 files come on different stw resolution. Only the stw in fba_level0 matches with the stw in ac_level0 (see Appendix how fba_level0 and ac_level0 matches).

All the level0 tables, except fba_level0, has a higher level (level1) table, where stw matches with ac_level0. Thus, it is simple to match level1 data and ac_level0 using stw in SQL queries, but other level0 data does not match.

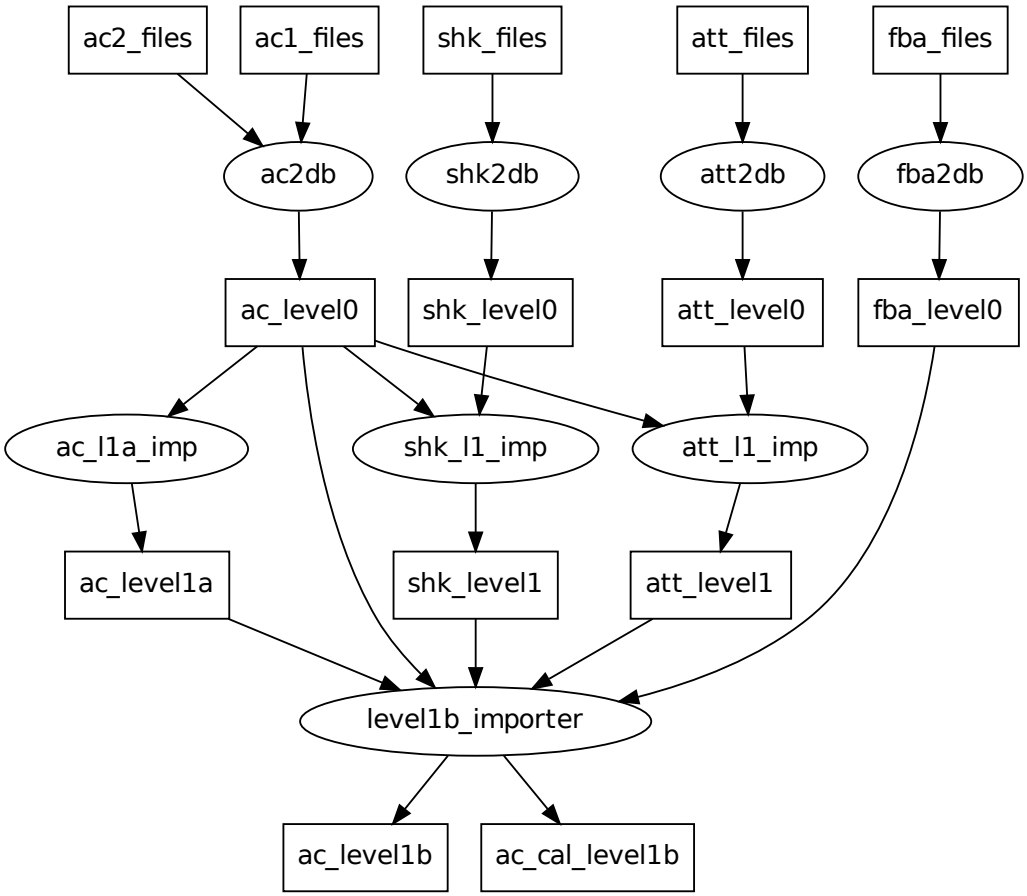

Figure 1: A dependency diagram of the processing and database system. The top layer represents level0 files. The names in the ovals represent executables that process and import data into database tables (named in the boxes below the top layer). The graph can be interpreted in the following way: att_l1_imp search for data in the att_level0 and ac_level0 database tables in order to ultimately import data into the att_level1 table, but is independent of all other processes in the figure.In the remaining text att_l1_imp= attitude_level1_importer, shk_l1_imp=shk_level1_importer, and ac_l1a_imp= ac_level1a_importer

Table columns

Table 1: ac_level0

Column	Type	Description
stw (key)	bigint	-
backend (key)	('AC1','AC2')	-
frontend	('549','495','572','555','SPL','119')	-
sig_type	('REF','SIG')	-
ssb_att	integer[]	-
ssb_fq	integer[]	the four lo freq. of the ssb modules
prescaler	integer	-
inttime	real	-
mode	integer	-
acd_mon	bytea, shape=(8,2),dtype='float64'	monitor data
cc	bytea, shape=(8,96),dtype='float64'	correlation coefficients

How to extract column data that has Type bytea and integer[] using python can be seen in the Appendix.

Table 2: fba_level0, to be used together with ac_level0 to find out what sig_type 'REF' means

Column	Type	Description
stw (key)	bigint	-
mech_type	('SK1','SK2','CAL')	-

Table 3: attitude_level0

Column	Type	Description
stw (key)	bigint	-
year	integer	-
mon	integer	-
day	integer	-
hour	integer	-
min	integer	-
secs	double precision	-
orbit	double precision	-
qt	double precision[]	qtarget (4-vector)
qa	double precision[]	qachieved (4-vector)
qe	double precision[]	qerror (3-vector)
gps	double precision[]	gps position and velocity (6-vector)
acs	double precision	-

Table 4: shk_level0

Column	Type	Description
stw (key)	bigint	-
shk_type (key)	('LO495','LO549','LO555','LO572',	lo freq.
	'SSB495','SSB549','SSB555','SSB572',	ssb tunings
	'mixC495','mixC549','mixC555','mixC572',	mixer current
	'imageloadB','hotloadA','hotloadB')	
shk_value	real	-

2.3 Level1 and level1a tables

attitude_level1, shk_level1, and ac_level1a contains processed level0 data. attitude_level1_importer, ac_level1a_importer, and shk_level1_importer are responsible for importing data into these tables. stw and backend are key-columns in these tables, and these key-columns matches among these tables.

Table columns

Table 5: attitude_level1

Column	Type	Description
stw (key)	bigint	-
backend (key)	('AC1','AC2')	-
mjd	double precision	-
lst	real	-
orbit	double precision	-
latitude	real	-
longitude	real	-
altitude	real	-
skybeamhit	integer	-
ra2000	real	-
dec2000	real	-
vsource	real	-
qtarget	double precision[]	-
qachieved	double precision[]	-
qerror	double precision[]	-
gpspos	double precision[]	-
gpsvel	double precision[]	-
sunpos	double precision[]	-
moonpos	double precision[]	-
sunzd	real	-
vgeo	real	-
vlsr	real	-
alevel	integer	-

Table 6: shk_level1

Column	Type	Description
stw (key)	bigint	-
backend (key)	('AC1','AC2')	-
lo	real	-
ssb	real	-
mixc	real	-
imageloadb	real	-
hotloada	real	-
hotloadb	real	-

Table 7: ac_level1a

Column	Type	Description
stw (key)	bigint	-
backend (key)	('AC1','AC2')	-
spectra	bytea, shape=(112*8,),dtype='float64'	spectrum in the frequency domain

2.4 Level1b tables

ac_level1b (contains target spectra) and ac_cal_level1b (contains sideband ratio spectrum and calibration spectrum) contains processed level0 and level1 table data. Level1b_importer (further described in Sect. 3) is responsible for importing data into these tables.

Table columns

Table 8: ac_level1b

Column	Type	Description
stw (key)	bigint	-
backend (key)	('AC1','AC2')	-
version (key)	integer	-
intmode (key)	integer	-
spectra	bytea,shape=(112*8,),dtype='float64'	calibrated spectrum
channels	integer	-
skyfreq	double precision	-
lofreq	double precision	-
restfreq	double precision	-
maxsuppression	double precision	-
tsys	real	-
sourcemode	('STRAT','ODD_H','ODD_N', 'WATER','SUMMER','DYNAM')	-
freqmode	integer	-
efftime	real	-
sbpath	real	-
calstw	bigint	stw of calibration spectrum

Table 9: ac_cal_level1b

Column	Type	Description
stw (key)	bigint	-
backend (key)	('AC1','AC2')	-
version (key)	integer	-
spectype (key)	('CAL','SSB')	-
intmode (key)	integer	-
spectra	bytea	shape=(112*8,),dtype='float64'
channels	integer	-
skyfreq	double precision	-
lofreq	double precision	-
restfreq	double precision	-
maxsuppression	double precision	-
sourcemode	('STRAT','ODD_H','ODD_N', 'WATER','SUMMER','DYNAM')	-
freqmode	integer	-
sbpath	real	

2.5 Importing data

Importing all data from level0 files into the level0 tables are done by the following commands:

```
$bin/ac2db /home/bengt/Downloads/02f8deef.ac1
$bin/ac2db /home/bengt/Downloads/02f8deef.ac2
$bin/att2db /home/bengt/Downloads/02f8a74e.att
$bin/shk2db /home/bengt/Downloads/02f8def1.shk
$bin/fba2db /home/bengt/Downloads/02f8def0.fba
```

Importing data into the three first level1 tables are done with the following commands:

```
$bin/shk_level1_importer
$bin/att_level1_importer
$bin/ac_level1a_importer
```

All data that is available will be imported into the different tables.

The calibration of ac_level1a data is then done by:

```
$bin/level1b_importer orbit backend
e.g.
$bin/level1b_importer 8575 AC1
```

Resulting data will be inserted in table ac_level1b (calibrated spectra) and ac_cal_level1b (tsys and ssb spectra). The old calibration procedure is performed based on data from one orbit, but ac_cal_level1b uses data from all scans that starts in an orbit. This means that there is a difference in which data that is considered in both the start and the end of the orbit (this is very easy to change). level1b_importer assumes that all data from the orbit is imported into the database, but nothing is clearly done if not enough data is found.

Extracting calibrated spectrum and data (in a similiar structure as the official level1b files) from the database into a hdf5 format file can be done by:

```
$bin/level1b_exporter orbit backend outputfile.hdf5
```

3 Calibration procedure

3.1 ac_level0 to ac_level1a

The processing of ac_level0 data (measured modified correlation coefficients) to ac_level1a (spectrum in the frequency domain) is partly described in [1], and is in short given here :
– correlation coefficients from the eight correlator chips with 96 lags each are individually approximately inverted to the true correlation (ρ) via Eq. 8 given in [1].
– a Hanning smoothing is applied on ρ
– an optimised fast Fourier transform algorithm is then applied: ρ is padded with zeros to be of length 112. the original sequence data[0] ... data[111] is replaced by data[0] ... data[111] 0.0 data[111] ... data[1], a sequence of length 224, which is a multiple of 32 that fits the applied Fourier transform. The resulting spectrum in the frequency domain is of length 112.

3.2 ac_level1a to ac_level1b

The old existing intensity calibration scheme is orbit based, although the reconstruction of LO-frequency due to temperature drifts is performed on an individual basis. The main steps of the intensity calibration scheme (v7) is here described:

- Discard spectra which have uncertain attitude data and data with tangent altitudes below 0 or above 120 km.
- Convert all measurements of the hot load into spectra of system temperature T_{sys}

  First measurements of the sky beam is interpolated to the stw of each hot load measurements. A $T_{sys,i}$ spectrum is then calculated as Eq. 4 in [1], i.e. :

  $$T_{sys,i} = \frac{c_i^S}{c_i^L - c_i^S}(T_L - T_S), \tag{1}$$

  where c_i^S and c_i^L are data from channel i of a spectrometer for the sky beam and hot load, respectively. T_L and T_S is the load and sky

brightness temperature, e.g.

$$T_S = \frac{hf_{sky}/k}{exp(hf_{sky}/k \cdot 2.7) - 1.0}, \tag{2}$$

where h and k is Planck's and Boltzmann's constants, respectively. It is then checked that T_{sys} values have reasonable values, and an average $< T_{sys} >$ spectrum of all good T_{sys} spectra is then used for the complete orbit in further steps.

- Transform all signal spectra into a antenna temperature scale:
  This is done in two steps, first without considering the baffle contribution. This is done by:

$$T'_{A,i} = \frac{(c_i^A - c_i^S) < T_{sys,i} >}{c_i^S} + T_S, \tag{3}$$

  where c_i^A is data from channel i of the target signal, and c_i^S is interpolated to the stw of the target signal.

- Determination of baffle contribution
  The baffle contribution T_{sp} is calculated as the median of the median of all spectra that are 10 km below or less of the maximum tangent altitude of all measurements. That is, T_{sp} is a scalar. The "correction" applied is then:

$$T_{A,i} = \frac{T'_{A,i} - T_{sp}}{\eta}, \tag{4}$$

  where

$$\eta = 1 - \frac{T_{sp}}{300}. \tag{5}$$

  Equations 3, 4, and 5 gives

$$T_{A,i} = \frac{1}{\eta}\left(\frac{(c_i^A - c_i^S) < T_{sys} >}{c_i^S} + T_S - T_{sp}\right). \tag{6}$$

  This is consistent with Equations 4,5, and 6 in [1].

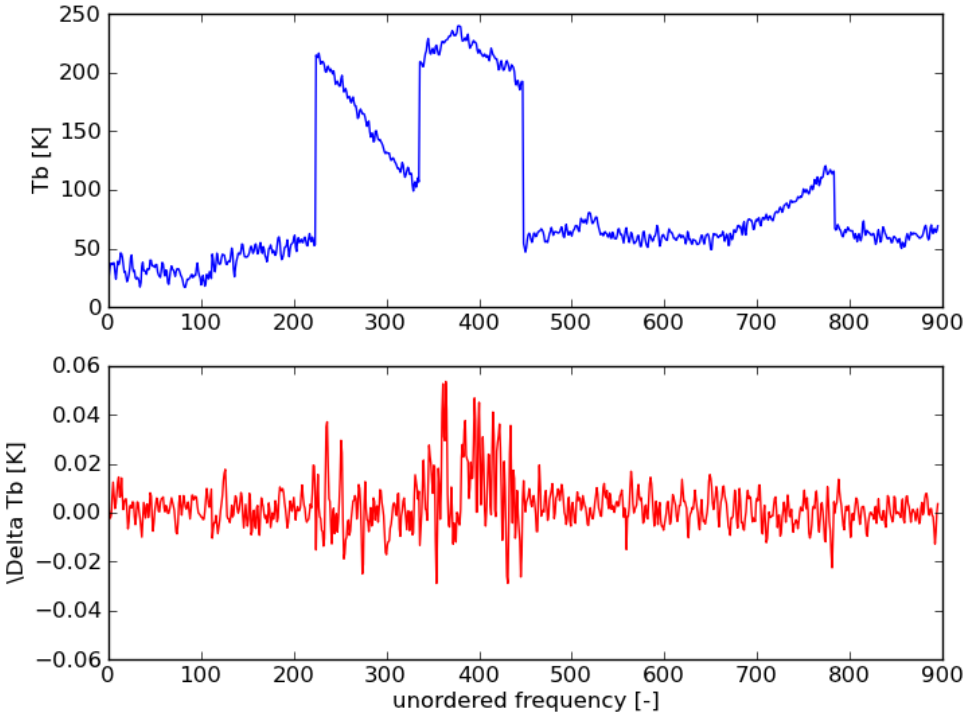

Figure 2: Upper panel: a level1b spectrum from AC1,STW=797932325, STRAT FM=2 created by the package described in this document, with a small modification, that is described in text. Lower panel: difference from the official level1b product (spectrum found in the file OB1B217F.HDF)

3.3 A short comparison

Figure 2 shows a comparison with a randomly selected intensity calibrated spectrum of the official level1b product. Differences are below 0.06 K, which we judged to be sufficiently small. For the match to be perfect, exactly the same data from the orbit must be used, since the actual Tsys spectrum used is the mean of all Tsys spectra from the orbit, and the baffle contribution is calulated from the median of the median of all high altitude spectra. Therefore in Figure 2 we have tried to do so by considering all data from the orbit and not all scans that starts in the orbit.

4 Further development

level1b_importer, which steers the level1b calibration, has been written to process data from scans that starts in the target orbit. The main function of this script starts basically by some SQL-queries to find out the start and end stw for the target orbit. With this information it is then easy to get all required data from the database tables. This means that to move away from the orbit based processing the main thing to change is the first queries.

The intensity calibration is performed by a python class named Level1b_cal that can be found in the file level1b_importer.py. This class has functions as for example:
– calibrate, that controls the calibration
– cleanref, that removes spurious cold sky spectrum
– cleancal, that removes spurious hot load spectrum
– gain, that calulates tsys spectrum
– medTsys, that returns a mean Tsys spectrum of all acceptable Tsys spectrum
calibrate1, that transform signal spectra into an antenna temperature scale.

Laboration with the calibration scheme should not be extremely difficult.

5 Required packages and installation

6 Appendix

Matching ac_level0 with fba_level0

One column in the ac_level0 table is named sig_type and tells if the signal is a measurement ('SIG') or reference ('REF'). 'REF' can be either a hot load or cold sky signal, and this is further specified in the fba_level0 table, which contains a column called mech_type. Thus, if sig_type is 'REF' one can join the two tables to find out the type of 'REF' signal. For example by the sql queries :
For ac2 data where the stw match is one to one

```
select stw, sig_type, mech_type from
ac_level0 natural join fba_level0
where sig_type='REF' and backend='AC2';
```

or for ac1 data where the stw is one stw unit ahead:

```
select ac_level0.stw, sig_type,
mech_type from
ac_level0 join fba_level0 on
(fba_level0.stw+1=ac_level0.stw) where
sig_type='REF' and backend='AC1';
```

Example how to extract column data that has Type bytea or integer[] using python

To extract data from database tables that have type bytea one must know the shape and datatype of the data. This information can be found in the tables in this document. Here is an example to extract data from the ac_level0:

```
import numpy
from pg import DB
class db(DB):
    def __init__(self):
        DB.__init__(self,dbname='odin')

con=db()
query=con.query('''select ssb_fq,cc from ac_level0''')
result=query.dictresult()
cc=numpy.ndarray(shape=(8,96),dtype='float64',
                 buffer=con.unescape_bytea(result[0]['cc']))
ssb_fq=eval(result[0]['ssb_fq'].replace('{','(').replace('}',')'))
```

Example how to identify scans

getscansac1 and getscansac2 are database functions that has been created in order to find out scans. The functions search for data in ac_level0 and fba_level0 and return stw and start of ac data, where start is the stw of the latest calibration measurement. The following SQL-queries show how these functions can be used, and the example also clarifies the meaning of a scan:

For AC2

```
select ac_level0.stw,start,backend,sig_type,mech_type,altitude
from ac_level0
natural join fba_level0
natural left join attitude_level1
natural join getscansac2()
where ac_level0.backend='AC2' and start>-1
order by ac_level0.stw;
```

For AC1

```
select ac_level0.stw,start,backend,sig_type,mech_type,altitude
from ac_level0
natural left join attitude_level1
natural join getscansac1()
join fba_level0 on (fba_level0.stw+1=ac_level0.stw)
where ac_level0.backend='AC1' and start>-1
order by ac_level0.stw;
```

and can result in someting like:

```
    stw     |   start    | backend | sig_type | mech_type | altitude
------------+------------+---------+----------+-----------+----------
  797607149 |  797607149 | AC2     | REF      | CAL       |
  797607213 |  797607149 | AC2     | SIG      | CAL       |  66394.2
  797607277 |  797607149 | AC2     | REF      | CAL       |
  797607341 |  797607149 | AC2     | SIG      | SK2       |  60728.1
  797607405 |  797607149 | AC2     | REF      | SK1       |
  797607469 |  797607149 | AC2     | SIG      | SK1       |  54946.1
  797607533 |  797607149 | AC2     | REF      | SK1       |
```

797607596	797607149	AC2	SIG	SK1	49403.8
			.		
			.		
			.		
797608300	797607149	AC2	SIG	SK1	16571.3
797608332	797607149	AC2	REF	SK1	
797608364	797607149	AC2	SIG	SK1	13635.4
797608396	797607149	AC2	REF	SK1	
797608428	797607149	AC2	SIG	SK1	10608.7
797608460	797607149	AC2	REF	SK1	
797608492	797608492	AC2	SIG	CAL	8020.76
797608524	797608492	AC2	REF	CAL	
797608556	797608492	AC2	SIG	CAL	10532.8
797608588	797608492	AC2	REF	CAL	

A scan is here defined as measurements between two hot load (CAL) measurements. These measurements are performed when the scanning turns, which can be seen in the altitude column. A scan is identified in the start column, which is the stw of the hot load measurement in the start of the scan. In this examlpe AC2 data with stw 797607149 to 797608460 belongs to scan 797607149.

References

[1] M. Ohlberg et al., *The Odin satellite II. Radiometer data processing and calibration*, Astronomy and astrophysics 402, L35–L38, 2003.