CHALMERS
Onsala Space Observatory

Working with Level 1b Odin Astronomy Data

Version Draft 0.1 of 2010-06-29, by Michael Olberg

Abstract

This document contains the necessary information for scientists to read and interpret Odin astronomy data at level 1b. A brief overview of the Odin data processing pipeline is presented, followed by a detailed description of the Odin data structures and the internal layout of Odin binary FITS tables. Tools to read the FITS tables and generate e.g. CLASS compatible data files are available.

Document: OSO/2010-001
Draft 0.1 of 2010-06-29
31 pages

Document approval

Prepared by: Michael Olberg Date: 2010-01-06

Checked by: Date:

Authorised by: Date:

Distribution

SNSB:
L. Nordh

SSC:
U. Frisk
S. Lundin

Project:

Version	Date	Change	Author
Draft 0.1	2000-01-01	first draft	Michael Olberg

Applicable Documents

Doc. ref		Title

Reference Documents

Doc. ref		Title
10.1051/0004-6361:20030335	RD01	The Odin satellite: II. Radiometer design and test
10.1051/0004-6361:20030336	RD02	The Odin satellite: II. Radiometer data processing and calibration

Contents

1 Introduction

This document tries to provide the necessary information for scientists who want to access publically available Odin astronomy data at level 1b from the data archive stored at the PDC Center for High Performance Computing at KTH, Stockholm. The information on how to get the necessary credentials to access the PDC data can be obtained elsewhere. *Need to add that info.*
In section 2 we summarize the internal workings of the Odin data processing pipeline and recall the various levels at which Odin data exist. Note, that this document is solely concerned with the retrieval of the final stage of the astronomy data processing, i.e level 1b.
During the various stages of the data processing, the pipeline software represents Odin data in memory in the form of a particular data structure called an OdinScan. This data structure is described in detail in section 3.
For archiving of astronomy data at level 1b OdinScans from one orbit are stored as the individual rows of a FITS binary table[1]. Some more information on these binary tables are presented in section 4.
A number of useful tools exist to extract individual OdinScan from the binary tables and store them in a more convenient format for the average user, e.g. standard FITS files readable by CLASS. These tools are described in section 4.1.

2 The Odin data processing pipeline

Processing of Odin data, both for the astronomy and aeronomy discipline, is typically done orbit by orbit and starts with gathering the necessary raw data files needed to generate the higher data levels. The raw data come in the following flavours:

SHK Housekeeping files which, most importantly contain the information on LO tuning and frontend configuration.

AC1 Raw science data files for the autocorrelator 1. Contains the raw correlator lags as well as the information on the spectrometer configuration, i.e. the bandwith/resolution which was used.

AC2 ditto for autocorrelator 2.

AOS Raw science data files for the acousto-optical spectrometer. Contains the power spectra as well as information on the AOS configuration. Spectra representing dark current readings or frequency calibration comb spectra are marked as such.

ATT The reconstructed timeline of satellite attitudes and orbit position and velocity. These files become available typically a couple of days after the actual observation, as the attitude processing is done a posteriori on ground.

Once the raw data are located, one OdinScan data structure (see section 3 below) is allocated and filled for each spectrum taken during the orbit in question. The filled data structures constitute Odin data level 0. Level 0 data are still unprocessed, e.g. correlator data are still represented as lags. However, from level 0 onwards no further access to the raw data files is necessary. The OdinScan data structure distinguishes between the following types of spectra:

SIG uncalibrated spectrum towards target (ON).

REF reference spectrum, i.e. an uncalibrated spectrum of cold sky taken towards a reference position during position switching.

CAL at level 0 the calibration spectrum towards the calibration load, at level 1 this is a spectrum of system temperature as a function of frequency.

CMB frequency comb spectrum (only for AOS.)

DRK CCD dark current spectrum (only AOS.)

SK1 a reference spectrum during sky switching using the sky 1 reference beam.

[1]For aeronomy the HDF format is used instead.

SK2 a reference spectrum during sky switching using the sky 2 reference beam.

All astronomy data were taken in either position switched mode or sky switched mode. At level 1 we (may) have in addition

SPE a calibrated spectrum.

SSB sideband ratio spectrum, only produced as part of the aeronomy pipeline.

AVE the average of several calibrated spectra.

Going from level 0 to level 1, the following steps are performed:

1. for the correlators, transform the raw correlator lags into a power spectrum. Keep track of which subbands were used with which correlator internal LO settings in order to generate the proper frequency scale.
2. for the AOS, analyse any frequency COMB spectra and use results for frequency calibration of subsequent science spectra.
3. calibrate the intensity scale of the individual SIG spectra, which subsequently will be labelled SPE. The intensity calibration typically uses all load measurements of one orbit in order to derive an average spectrum of system temperature. Reference (or sky beam) spectra are properly interpolated onto the times of the SIG spectra. Details of the calibration are given in RD02.

The software for the Odin data processing pipeline consists of (1) a collection (i.e. a library) of C routines, for effective access to raw data and optimized code for the handling of Odin data structures, and (2) interfaces to higher level scripting languages (originally Perl was used, later Python) to allow calls to this C library and combining these calls into the various steps needed to arrive at level 1b.

3 Detailed description of the Odin data structure

The C header file defining the Odin data structure OdinScan is very heavily commented and should be self-explanatory. It's listing is reproduced in appendix A. The OdinScan structure consists of a header section of 408 bytes, followed by up to 1728 (the length of an AOS spectrum) channels. Correlator spectra typically have 896 channels. Each data channel is represented as a single precision floating point number, i.e as four bytes. Thus the maximum number of bytes needed to store one Odin spectrum is $408 + 1728 * 4 = 7320$.

For the convenience of the reader we here present additional comments on the various data items in the header section:

Table 1: Layout of the OdinScan structure, size is given in bytes.

name	type	size	description
Version	unsigned short	2	Contains the version number of the OdinScan structure that was in use when the data were generated. If printed in hexadecimal notation, the first nibble gives the major number and the second nibble the minor number. The version history is found at the beginning of the file listing in appendix A. At present its value is 0x0106, i.e. version 1.6.
Level	unsigned short	2	Indicates the level and version numbers of applied calibration procedures and pointing constants. The upper nibble in hexadecimal notation indicates the version of ACS alignment constants that were in use. This data word can typically be ignored by the end user.

Continued on next page

name	type	size	description
Quality	unsigned long	4	A 32 bit long word were each bit represents a possible status, warning or error flag. Please see the file listing in appendix A for a complete list. Please note that the lowest four bits should be interpreted as a reset counter for the satellite time word (STW). As such a reset never happened during the life time of Odin, all data up to the summer of 2009 will have a bit pattern of 0000 here, whereas later data have 0001 due to the roll-over of the STW at that time.
STW	unsigned long	4	The satellite time word, a 32 bit long integer keeping time onboard the Odin satellite, ticking at 16 Hz. As this is stored when data were transferred from the spectrometers to the mass memory, it corresponds to the end of the measurement.
MJD	double	8	The modified Julian date of observation, corresponding to the STW above.
Orbit	double	8	The orbit number at the start of the observation. The fractional part is the phase (position) within the orbit relative to the equator crossing.
LST	float	4	The local (mean) sidereal time at the start of the observation in seconds. It is calculated based on UTC and the current satellite position.
Source	char[32]	32	A string of 32 characters holding the name of the source, which is picked up from the unvalidated time line (block 1) of an attitude file. As such this is the name of the source given by the planners, and do not necessarily follow standard naming conventions for astronomical sources.
Discipline	short	2	For astronomy this word should contain the value 2, whereas the value 1 indicates aeronomy data.
Topic	short	2	The various astronomy topical teams are coded as follows SOLSYS 1 solar system STARS 2 late type stars EXTGAL 3 extragalactic LMC 4 Magellanic clouds PRIMOL 5 primordial molecules SPECTR 6 spectral scans CHEM 7 interstellar chemistry GPLANE 8 galactic plane GCENTR 9 galactic centre GMC 10 giant molecular clouds SFORM 11 star formation DCLOUD 12 dark clouds SHOCKS 13 shocks and outflows PDR 14 photon dominated regions HILAT 15 high latitude clouds ABSORB 16 absorption line studies ORION 17 common Orion project CALOBS 18 calibration observations COMMIS 19 commissioning phase measurement
Spectrum	short	2	An integer counting the individual spectra of one orbit.

Continued on next page

name	**type**	**size**	**description**
ObsMode	short	2	The observing mode in use when this spectrum was taken. The only modes that were used during the Odin mission are TPW (total power, i.e. position switch) coded as the value 1, and SSW (sky switch) coded as 2.
Type	short	2	The type of the spectra as described above in section 2. But see the file listing in appendix A for further details.
Frontend	short	2	The frontend (receiver) used for this observation, coded as REC_555 1 mixer B2 REC_495 2 mixer A2 REC_572 3 mixer B1 REC_549 4 mixer A1 REC_119 5 mixer C REC_SPLIT 6 power combiner, not used in astronomy
Backend	short	2	The backend (spectrometer) used for this observation, coded as AC1 1 correlator 1 AC2 2 correlator 2 AOS 3 acousto optic spectrometer FBA 4 3 channel filter bank (aeronomy)
SkyBeamHit	unsigned short	2	The 16 bits of this word indicate possible hits of major sources of submillimetre emission, like the Sun, Moon, Jupiter and Saturn, by one of the skybeams or the main beam. See the file listing in appendix A for further details.
RA2000	float	4	The right ascension (J2000) of the source (direction of pointing) in degrees. Calculated from Qachieved below.
Dec2000	float	4	The declination (J2000) of the source (direction of pointing) in degrees. Calculated from Qachieved below.
VSource	float	4	For astronomy, the velocity of the source with respect to the local standard of rest (LSR) in m/s. It is extracted from the unvalidated time line (block 1) of an attitude file, i.e. a number which was input by the schedule planners.
u	point	12	This is the only member of the structure which is not represented by a fundamental C variable type, but rather as a union, holding three floating point variables with different meanings depending on the discipline. See its definition in the code listing of appendix A at lines 368–381. For astronomy, map offsets and map tilts can be stored here, but the pipeline will typically leave these values equal to zero.
Qtarget	double[4]	32	The reference attitude given as a quaternion (4-vector) as listed in columns 8-10 of block 3 of an attitude file.
Qachieved	double[4]	32	The achieved attitude given as a quaternion (4-vector) as listed in columns 11-14 of block 3 of an attitude file.
Qerror	double[3]	24	The attitude error in degrees, describing the pointing uncertainty around the 3 principle axes of the satellite. Listed in columns 15-17 of block 3 of an attitude file.
GPSpos	double[3]	24	The geocentric position X,Y,Z in meter of the satellite at the end of the observation as reported by the GPS receiver onboard of Odin and as listed in columns 18-20 of block 3 of an attitude file.

Continued on next page

name	type	size	description
GPSvel	double[3]	24	The geocentric velocity $\dot{X}$, $\dot{Y}$, $\dot{Z}$ in meter per second of the satellite at the end of the observation as reported by the GPS receiver onboard of Odin and as listed in columns 21-23 of block 3 of an attitude file.
SunPos	double[3]	24	The geocentric position of the Sun in meter. Calculated based on current time using a low precision ephemeris. Accuracy is guaranteed to be one arcmin or better.
MoonPos	double[3]	24	The geocentric position of the Moon in meter. Calculated based on current time using a low precision ephemeris. Accuracy is guaranteed to be one arcmin or better.
SunZD	float	4	The solar zenith angle in degrees. Calculated from the position of the Sun and therefore with the same intrinsic accuracy.
Vgeo	float	4	The velocity of the satellite with respect to the Earth in meter per second. Calculated from current time and satellite GPS position and velocity.
Vlsr	float	4	In astronomy, the velocity of the satellite with respect to the local standard of rest (LSR) in meter per second. Calculated from current time and satellite GPS position and velocity.
Tcal	float	4	The Rayleigh Jeans temperature in Kelvin of the calibration load used during the intensity calibration of this spectrum. The physical temperature of the load is retrieved from level 0 house keeping data files.
Tsys	float	4	The mean value of the system temperature in Kelvin used during the intensity calibration of this spectrum. Calculated during calibration. Note, that this is an average value of the system temperature, the CAL spectra at level 1b contain a channel by channel calculated system temperature.
SBpath	float	4	The path length in meter of the SSB diplexer. Extracted from house keeping data files.
LOFreq	double	8	The (first) local oscillator frequency in Hz used for this observation in the rest frame of the satellite. Calculated from the information on HRO and PRO frequency stored in level 0 house keeping data files.
SkyFreq	double	8	The frequency in Hz of the centre channel in the rest frame of the satellite, i.e not Doppler corrected. Calculated from the LO frequency and the value of the IF frequency, which will be either +3900 MHz or -3900 MHz. The sign is calculated based on the setting of the SSB tuning mechanism, stored in the level 0 house keeping data files. Note that the index of the centre channel `cc` is defined as `cc = spectrum->Channels/2;` with all the implications of an integer division. This means, that for an odd number of channels, the division yields the correct array index of the centre channel, for an even number of channels we get the index of the channel just above the centre of the band (which falls between two channels).

Continued on next page

name	type	size	description
RestFreq	double	8	The frequency in Hz of the centre channel in the rest frame of the observed object, i.e Doppler corrected. The same definition of centre channel applies as for the previous member. Calculated from sky frequency and applicable Doppler correction (see Vlsr). The frequency correction is calculated using the radio convention: $$f_{sky} = f_{rest} \cdot \left(1 - \frac{v}{c}\right)$$ Where v is the sum of the source velocity and the LSR velocity and the calculated LSR velocity of the satellite.
MaxSuppression	double	8	The sideband frequency in Hz corresponding to the frequency of maximum suppression in the rest frame of the satellite, i.e not Doppler corrected.
SodaVersion	double	8	The version number of the attitude reconstruction software (SODA) used at SSC during production of attitiude files. Note, that a double variable is used here because of historical reasons. In earlier version of this structure the amount of frequency throw was supposed to be stored here. As FSW observations were never done with Odin, this field was available for this new purpose. In order not to change the size of the sturcture, the data type was kept.
FreqRes	double	8	The spacing in Hz between neighbouring channels for this spectrum.

FreqCal	double[4]	32	Frequency calibration coefficients in Hz for the spectrometer in use.
			For the correlators, these are the four local oscillator frequencies of the SSB modules, in the order in which individual bands are read out from the instrument, i.e. 1,3,2,4 when compared to Omnisys documentation. Typical values:
			`FreqCal[0] =  3600.0e+06`
			`FreqCal[1] =  3800.0e+06`
			`FreqCal[2] =  4000.0e+06`
			`FreqCal[3] =  4200.0e+06`
			For the acousto optic spectrometer, these are the fit results from the last frequency comb measurement performed. Typical values:
			`FreqCal[0] =  2.100e+09`
			`FreqCal[1] =  6.200e+05`
			`FreqCal[2] =  4.000e+00`
			`FreqCal[3] = -1.000e-02`

Continued on next page

name	type	size	description
IntMode	int	4	The integration mode of the backend during the observation. The information is retrieved from values reported by the spectrometers. For the AOS these are AOS_LONG 1 1728 32-bit data AOS_SHORT 2 1728 16-bit data AOS_HALF 3 864 32-bit data AOS_FOUR 4 432 32-bit data AOS_CENTRE 5 high resolution at centre AOS_WINGS 6 high resolution in wings AOS_WINDOW 7 high resolution window For the correlators AC1 and AC2 these are AC_XHIRES 1 1 x 100 MHz at 125 kHz resolution AC_HIRES 2 2 x 100 MHz at 250 kHz resolution AC_MEDRES 3 4 x 100 MHz at 500 kHz resolution AC_LOWRES 4 4 x 200 MHz at 1000 kHz resolution AC_YHIRES 5 1 x 100 MHz at 1000/7 kHz resolution For the correlators in split mode (not used in astronomy) bits 4 and 5 can be set, having the following meaning AC_SPLIT bit 4 correlator split between frontends. AC_UPPER bit 5 correlator upper half.
IntTime	float	4	The integration time in seconds, i.e. the duration of this observation.
EffTime	float	4	The effective integration time in seconds, i.e. one will get the noise level in this spectrum by using this time and the system temperature from above and the usual radiometer formula: $\Delta T = \frac{T_{sys}}{\sqrt{\Delta\nu \cdot t}}$ where $\Delta\nu$ is the FreqRes from above.
Channels	int	4	The number of channels in this spectrum. The maximum number of channels that may occur is given by the number of channels in the AOS, i.e. 1728.
data	float[*]		The array holding the actual spectrum. Its size in bytes varies and is given by the number stored in the previous item (Channels) times 4.

4 Odin binary FITS tables

The binary FITS tables produced and stored as level 1b simply use the FITS standard to store a table where each row is a data structure as defined in 3, i.e. use the standard FITS keywords for type, form and unit to encode each structure member of OdinScan as a column of a FITS binary table.
Level 1b files stored at PDC follow the following naming convention:
`O`x`1B`$yyyy$`.FIT.gz`
where x is a letter encoding the backend (`A` = AC1, `B` = AC2, `C` = AOS) and $yyyy$ is the orbit number expressed as a 4-digit hexadecimal number. The first letter of the filename will be `O` to indicate its origin at Onsala Space Observatory, the 3rd and 4th letter confirm that this is a data file at level 1b.

4.1 Tools

The easiest way to analyse the Odin data is to extract the individual OdinScan structures from a FITS binary table, once this has been retrieved from the PDC, and convert the structures into a format suitable for the data reduction program of choice. The software to do this is available as source code from a subversion repository maintained by the Odin aeronomy group at Chalmers Technical University. Read only access is possible without username or password:

```
$ svn checkout http://svn.rss.chalmers.se/svn/odinsmr/oops/trunk oops
```

This command will retrieve the current version into a subdirectory `oops`[2]. In there you'll find a number of subdirectories:

Library: contains the low level C-routines used by most Odin related programs and scripts. On a Linux platform building the software library will require

```
$ ./configure
$ make
```

Programs: contains utilities based on the previous library, i.e. you need to build the library described above first, before you can compile and link any of the software in this directory. Here you'll find programs `rfits` and `odin2fits` which allow you to extract all spectra from an Odin binary FITS table and convert each of them to the more standard, single dish FITS files which e.g. CLASS[3] can understand, respectively. Again a `configure` file is available to generate a Makefile for your system:

```
$ ./configure
$ make rfits
$ make odin2fits
```

Please note, however, that before you can build these programs you need to have the `cfitsio` library from NASA's HEASARC[4] installed on your system before you can build these programs. This library is typically available as a ready-made package for most Linux distributions, e.g. on Debian (Ubuntu) this would involve installing package `libcfitsio3-dev` and its dependencies, or `cfitsio-devel` on rpm-based distributions like Red Hat or SuSE. Else you may want to download and build that library following the instructions on the HEASARC site, in this case you will have to adapt the Makefile for building the Odin software.

Once you have the two programs (`rfits, odin2fits` in place you may use them to translate a file downloaded from PDC into a CLASS compatible format following the following recipe. Let's assume that the file you downloaded from the PDC is called `OC1B9A12.FIT.gz`, i.e. AOS level 1b data for orbit 0x9A12 (hex) = 39442 (decimal).

```
$ gunzip OC1B9A12.FIT.gz
$ rfits -file OC1B9A12.FIT
```

This will create a (large) number of binary dumps of OdinScan structures in your current directory, each representing one AOS spectrum. The files will follow the naming convention

`AOS.`*xxxxxxxx*`.`*yyy*

where *xxxxxxxx* is a hexadecimal representation of the satellite time word when this spectrum was observed, and *yyy* is the type of the spectrum, which at level 1b is either `SPE` for a calibrated spectrum or `CAL` for a spectrum of the system temperature.

Each of these can then be turned into a single dish FITS file readable by CLASS via

[2] oops = Onsala Odin Processing System
[3] `http://iram.fr/IRAMFR/GILDAS/`
[4] `http://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html`

```
$ ls -1 AOS.*.??? > spec.list
$ odin2fits -list spec.list
```

After this operation there will be a FITS file for every spectrum that was listed in `spec.list`. The file names of the FITS files will be the same as those of the original binary dumps, but with the original extension replaced by `.fits`. Please refer to CLASS documentation on how to read these into the CLASS system.

Instead of working with CLASS you may chose to work with the binary dumps directly, this can be done either with Python or MatLab (and compatible software like Octave). The necessary software is found in the following subdirectories.

Python: This subdirectory contains scripts used by the Odin data processing pipeline. These will probably not be too useful for the general user as they require a certain infrastructure in the form of SQL database tables to be present. However, here you will also find a Python module allowing you to work with OdinScan structures inside Python. Assuming that your Linux system has the necessary packages installed (package `python-dev`), building the module requires the following two steps (the second one needs to be carried out with root priviliges):

```
$ python setup.py build
$ python setup.py install
```

Once the Odin pythin module is available a spectrum can be read into python using code like the following:

```python
from pylab import *  # use e.g. matplot lib for plotting
import odin
s = odin.Spectrum("AOS.12345678.SPE")
print s              # will show extensive header info
d = s.data           # get spectrum data
f = s.frequency()    # generate a frequency vector for the data
plot(f, d)           # plot data against frequency
show()
```

MatLab: This subdirectory contains MatLab routines (these will work with the free MatLab alternative Octave, too) to read, and work with, the individual OdinScan structures which are written to disk if you run `rfits` on a Odin binary FITS table. The same example as listed under the python section would now read

```
s = loadspec("AOS.12345678.SPE");
d = s.data           # get spectrum data
f = frequency(s)     # generate a frequency vector for the data
plot(f, d)           # plot data against frequency
```

A Listing of `odinscan.h`

```c
/**@name OdinScan */

#ifndef ODINSCAN_H
#define ODINSCAN_H

/* $Id: odinscan.h,v 1.3 2007/02/01 07:11:23 olberg Exp $ */

/**@name OdinScan

     This structure plays a central role for the processing
     of Odin SMR spectral line data. It is in this format that data will
     be stored and handled by the processing computer at Onsala Space
     Observatory. Many of the routines described further down in this
     documentation require a pointer to such a structure as one of their
     input and/or output parameters.

     History:

     Version 0.1:
     Original version sent out to members of Odin data working group for
     discussion on March 19, 1997.

     Version 0.2:
     Slightly modified version presented at Odin data working group
     meeting in Toulouse, June 5, 1997.

     Version 0.3:
     Incorporated changes suggested at Odin data working group meeting
     in Toulouse, June 5, 1997 and rearranged and grouped structure
     members following suggestions made by Alain Lecacheux.

     Version 0.4:
     Added source name and topic.
     Added positions of Sun and Moon.
     Added flags indicating if sky beams point at Sun, Moon, Earth
     or avoidance zone.
     Based on agreements from Odin meetings on April 20, 1998 at CTH
     and on May 15, 1998 at SSC.

     Version 0.5:
     Added sideband IF frequency which has maximum suppression.
     Added solar zenith angle (requested by Eric Le Flochmoen, France).
     Based on e-mail received from Frank Merino, MISU.

     Version 0.6:
     Rearrangements of a few structure members in order to optimize for
     conversion to FITS binary tables.

     Version 0.7:
     Major changes concerning new and removed structure members, following
     Odin scheduling meeting at SSC, Stockholm on March 9, 1999.
     Units of angles changed from radian to degrees.

     Version 0.8:
     Added structure member EffTime, following calibration meeting at MISU.
```

```
    Removed structure member Bandwidth. Added union to support tangent point
    in aeronomy mode and map position in astronomy mode.

    Version 0.9:
    Added new spectra types: SK1, SK2, DRK for sky beam 1, sky beam 2
    and AOS dark spectrum, respectively.
    Edited comments for frontend types.

    Version 1.0:
    Added backend FBA, after decision at OST 35 to store filter bank data
    as 3 channel spectrometer and abandoning extra file formats for FBA data.
    Added flag SATURNMB and replaced flags EAC1, EAC2 and EAOS by ECALMIRROR
    and ESIGLEVEL.
    Replaced member Vhel by Vgeo, following suggestion by Nicolas Biver.
    Added doc++ style comments.

    Version 1.1
    Added spectrum type SPE for calibrated spectra. Added bit definition
    WBANDADJUST which will be set in correlator spectra which show
    stair-caseing. Redefined bit definition for ICOMMISSION.

    Version 1.2
    Replaced structure member 'Trec' by 'SBpath', after discussion during
    a meeting of the Odin SMR aeronomy data retrieval group at Onsala, May
    21-23, 2001.
    Added definitions of constants for storing the aeronomy mode in member
    'Topic'.

    Version 1.3
    Added spectrum type SSB for sideband ratio, after SMR-RG meeting in
    Bordeaux. Also added type AVE for averaged spectra, mainly meant for
    astronomy.

    Version 1.4
    Corrected definitions of SkyBeamHit related constants (EARTH1, ...)
    to correspond to individual bits.

    Version 1.5
    Introduced new way of describing IntMode for AC1 and AC2. These new
    mode designations are characterised by setting bit ADC_SEQ in header
    variable IntMode. Bits AC_SPLIT and AC_UPPER are replaced by ADC_SPLIT
    and ADC_UPPER. All IntMode dependent code is supposed to handle both
    old and new modes and should check bit ADC_SEQ before processing.

    Version 1.6
    Replaced header variable FreqThrow by SodaVersion, in order to store
    software version used during attitude reconstruction.
*/

//@{
/**@name Constants

   The following macros should be used whenever possible rather than
   integer constants, to make sure that program code stays portable
   from one version to another. This means you should use code like
   \begin{verbatim}
   if (scan->Backend == AOS) ...
```

```
\end{verbatim}
rather than
\begin{verbatim}
if (scan->Backend ==    3) ...
\end{verbatim}
to test for the backend information in an OdinScan structure.

The following constants (macros) are defined:
\begin{verbatim}

#define ODINSCANVERSION  current version

Define the disciplines:
#define AERO             aeronomy
#define ASTRO            astronomy

Define the various topics in astronomy:
#define SOLSYS           solar system
#define STARS            late type stars
#define EXTGAL           extragalactic
#define LMC              Magellanic clouds
#define PRIMOL           primordial molecules
#define SPECTR           spectral scans
#define CHEM             interstellar chemistry
#define GPLANE           galactic plane
#define GCENTR           galactic centre
#define GMC              giant molecular clouds
#define SFORM            star formation
#define DCLOUD           dark clouds
#define SHOCKS           shocks and outflows
#define PDR              photon dominated regions
#define HILAT            high latitude clouds
#define ABSORB           absorption line studies
#define ORION            common Orion project
#define CALOBS           calibration observations
#define COMMIS           commissioning phase measurement

Define modes for aeronomy:
#define STRAT            stratospheric mode (1 or 2)
#define ODD_N            odd nitrogen mode
#define ODD_H            odd hydrogen mode (1,2 or 3)
#define WATER            water mode (1 or 2)
#define SUMMER           summer mesosphere
#define DYNA             dynamic ...

Available backends:
#define AC1              correlator 1
#define AC2              correlator 2
#define AOS              acousto optic spectrometer
#define FBA              3 channel filter bank

Backend modes:
#define AOS_LONG         1728 32-bit data.
#define AOS_SHORT        1728 16-bit data.
#define AOS_HALF          864 32-bit data.
#define AOS_FOUR          432 32-bit data.
#define AOS_CENTRE       high resolution at centre.
```

```
#define AOS_WINGS        high resolution in wings.
#define AOS_WINDOW       high resolution window.

#define AC_XHIRES        1 x 100 MHz at  125 kHz resolution.
#define AC_HIRES         2 x 100 MHz at  250 kHz resolution.
#define AC_MEDRES        4 x 100 MHz at  500 kHz resolution.
#define AC_LOWRES        4 x 200 MHz at 1000 kHz resolution.
#define AC_YHIRES        1 x 100 MHz at 1000/7 kHz resolution (seven chips!)
#define AC_SPLIT         correlator split between frontends.
#define AC_UPPER         correlator upper half.
#define ADC_SEQ          new version for ADC mode coding
#define ADC_SPLIT        correlator split between frontends (new)
#define ADC_UPPER        correlator upper half (new)

Available receivers (frontends):
#define REC_555          mixer B2.
#define REC_495          mixer A2.
#define REC_572          mixer B1.
#define REC_549          mixer A1.
#define REC_119          mixer C.
#define REC_SPLIT        power combiner, correlators only.

Define the various observing modes:
#define TPW              total power measurement.
#define SSW              sky switching.
#define LSW              load switching.
#define FSW              frequency switching (only for 119 GHz.)

Define the various spectrum types:
#define SIG              on spectrum towards target.
#define REF              reference spectrum.
#define CAL              calibration spectrum towards load.
#define CMB              frequency comb spectrum (only for AOS.)
#define DRK              CCD dark current spectrum (only AOS.)
#define SK1              sky 1 reference beam.
#define SK2              sky 2 reference beam.
#define SPE              calibrated spectrum.
#define SSB              sideband ratio spectrum.
#define AVE              averaged calibrated spectra.

Define flags for avoidance objects and zones:
#define SKYBEAMS         number of sky beams.
#define EARTH1           sky beam 1 points at Earth.
#define MOON1            sky beam 1 points at Moon.
#define GALAX1           sky beam 1 points at galactic plane.
#define SUN1             sky beam 1 points at Sun.
#define EARTH2           sky beam 2 points at Earth
#define MOON2            sky beam 2 points at Moon
#define GALAX2           sky beam 2 points at galactic plane
#define SUN2             sky beam 2 points at Sun.
#define EARTHMB          main beam points at Earth.
#define MOONMB           main beam points at Moon.
#define JUPITERMB        main beam points at Jupiter.
#define SATURNMB         main beam points at Saturn.

A few useful macros to determine the size of the structure below:
#define SOURCENAMELEN    length of source name.
```

```
    #define MAXCHANNELS     maximum number of channels.
    #define MAXDATA         maximium length of data.
    #define SCANLEN         scan length in bytes.
    #define HEADLEN         length of header in bytes.

    Error, warning and info codes (preliminary):
    #define STWRSTMASK      nibble will hold reset cnt.
    #define EPLATFORM       platform error.
    #define EPLL            PLL error, LO unlocked.
    #define ECALMIRROR      position of cal mirror undefined
    #define ESIGLEVEL       signal level in error
    #define WFREQUENCY      warn: frequency unreliable.
    #define WAMPLITUDE      warn: amplitude unreliable.
    #define WPOINTING       warn: unreliable pointing
    #define WBANDADJUST     one or more correlator bands adjusted
    #define ILINEAR         spectrum has been liearised
    #define ISORTED         spectrum has been frequency sorted
    #define ICOMMISSION     taken during commissioning.
    \end{verbatim}
*/

#define ODINSCANVERSION 0x0106

/*
    Note, that in the various lists below enumerations start at 1
    and not(!) at 0.
    This enables proper treatment of uninitialized structure members
*/
#define UNDEFINED   0

#define AERO        1
#define ASTRO       2

#define SOLSYS      1
#define STARS       2
#define EXTGAL      3
#define LMC         4
#define PRIMOL      5
#define SPECTR      6
#define CHEM        7
#define GPLANE      8
#define GCENTR      9
#define GMC        10
#define SFORM      11
#define DCLOUD     12
#define SHOCKS     13
#define PDR        14
#define HILAT      15
#define ABSORB     16
#define ORION      17
#define CALOBS     18
#define COMMIS     19

#define STRAT       1
#define ODD_N       2
#define ODD_H       3
#define WATER       4
```

```
#define SUMMER        5
#define DYNA          6

#define AC1           1
#define AC2           2
#define AOS           3
#define FBA           4

#define AOS_LONG      1
#define AOS_SHORT     2
#define AOS_HALF      3
#define AOS_FOUR      4
#define AOS_CENTRE    5
#define AOS_WINGS     6
#define AOS_WINDOW    7

#define AC_XHIRES     1
#define AC_HIRES      2
#define AC_MEDRES     3
#define AC_LOWRES     4
#define AC_YHIRES     5
#define AC_SPLIT  (1<<4)
#define AC_UPPER  (1<<5)

#define ADC_SEQ     (1<<8)
#define ADC_SPLIT   (1<<9)
#define ADC_UPPER   (1<<10)

#define REC_555       1
#define REC_495       2
#define REC_572       3
#define REC_549       4
#define REC_119       5
#define REC_SPLIT     6

#define TPW    1
#define SSW    2
#define LSW    3
#define FSW    4

#define SIG    1
#define REF    2
#define CAL    3
#define CMB    4
#define DRK    5
#define SK1    6
#define SK2    7
#define SPE    8
#define SSB    9
#define AVE   10

#define SKYBEAMS          2
#define EARTH1       0x0001
#define MOON1        0x0002
#define GALAX1       0x0004
#define SUN1         0x0008
#define EARTH2       0x0010
```

```
#define MOON2         0x0020
#define GALAX2        0x0040
#define SUN2          0x0080
#define EARTHMB       0x0100
#define MOONMB        0x0200
#define JUPITERMB     0x0400
#define SATURNMB      0x0800

#define SOURCENAMELEN    32
#define MAXCHANNELS    1728
#define MAXDATA MAXCHANNELS*sizeof(float)
#define SCANLEN sizeof(struct OdinScan)
#define HEADLEN (SCANLEN-MAXDATA)

#define STWRSTMASK    0x0000000f
#define EPLATFORM     0x00000010
#define EPLL          0x00000020
#define ESIGLEVEL     0x00000100
#define ECALMIRROR    0x00000800
#define WFREQUENCY    0x00001000
#define WAMPLITUDE    0x00002000
#define WPOINTING     0x00004000
#define WBANDADJUST   0x00010000
#define ILINEAR       0x01000000
#define ISORTED       0x02000000
#define ICOMMISSION   0x10000000

typedef union {
  /* aeronomy: tangent point parameters */
  struct {
    float  Longitude;
    float  Latitude;
    float  Altitude;
  } tp;
  /* astronomy: map parameters */
  struct {
    float  Xoff;
    float  Yoff;
    float  Tilt;
  } map;
} point;

/**@name Layout */
struct OdinScan {
  /** Version number.
      The version number of {\tt odinscan.h} which was in use when
      the data were created. When reading a spectrum from disk, one should
      check that the version stored in the spectrum header agrees with
      the version your software was compiled with:
      \begin{verbatim}
      if (spectrum->Version == ODINSCANVERSION) {
         // ok to process with this software
         ...
      }
      \end{verbatim}
      When writing data the software should make sure that the version
      number is set correctly.
```

```
        \begin{verbatim}
        spectrum->Version = ODINSCANVERSION;
        \end{verbatim}
        When read as a hexadecimal number, the high nibble will contain the
        major version number, the low nibble the minor version. During
        software development the major version number will be 0 and any
        software using major version 0 shall be considered preliminary.

        Related constant: ODINSCANVERSION
    */
    unsigned short Version;

    /** Level of data reduction applied.
        This word is used to indicate level and version numbers of
        calibration procedures and pointing constants have been used.
    */
    unsigned short Level;

    /** Status info for platform and payload.
        Up to 32 bit of status information for various platform and payload
        components. Macros (see below) starting with {\tt E} indicate
        error conditions, those starting with {\tt W} indicate warnings and
        those starting with {\tt I} are for informative purposes.

        Related constants:

        STWRSTMASK,
        EPLATFORM,
        EPLL,
        EAOS,
        EAC1,
        EAC2,
        WFREQUENCY,
        WAMPLITUDE,
        WBANDADJUST,
        ICOMMISSION

        @see Constants
    */
    unsigned long Quality;

    /** Satellite time word.
        This is simply a copy of the satellite time word from the first
        block of level 0 data belonging to this spectrum. As this is the
        STW when data were transferred from the spectrometers to the mass
        memory, it corresponds to the end of the measurement.
     */
    unsigned long STW;

    /** Modified Julian date of observation.
        The modified Julian date at the start of the observation.
        The modified Julian date is related to the Julian date {\tt JD}
        in use in astronomy by:
        \begin{verbatim}
        JD = spectrum->MJD + 2400000.5;
        \end{verbatim}
        Note, that the Julian date starts at noon, whereas the modified
```

```
        Julian date starts at midnight, e.g. MJD=0.0 corresponds to 1858
        November 17 at 0.0 UTC. The fractional part gives UTC in seconds
        of day via
        \begin{verbatim}
        UTC = modf(spectrum->MJD)*86400.0;
        \end{verbatim}
    */
    double MJD;

    /** Number of orbit plus fraction.
        The orbit number at the start of the observation. The fractional part
        is the phase (position) within the orbit relative
        to the equator crossing. The orbit number is extracted from
        column 6 of block 3 of an attitude file.
    */
    double Orbit;

    /** Local sidereal time of observation.
        The local (mean) sidereal time at the start of the observation in
        seconds. It is calculated based on UTC and the current
        satellite position. The latter in turn is derived from the GPS
        position retrieved from columns 18-20 of block 3 of an attitude file.
    */
    float  LST;

    /** Source name.
        A string of 32 characters holding the name of the source
        (in astronomy) or some other description of the current spectrum
        (in aeronomy).

        Related constants:

        SOURCENAMELEN

        The source name is extracted from the unvalidated time line (block 1)
        of an attitude file. The source name will be stored (with 16 characters
        only) in comment fields enclosed in square brackets on the same line as
        the {\tt newompb} keyword. Example:
        \begin{verbatim}
        newompb:    0  [CALOBS W3(OH)              -45.0 POS         ]
                                ________________
        \end{verbatim}
    */
    char     Source[SOURCENAMELEN];

    /** Discipline.
        The discipline for which this spectrum was taken, either aeronomy or
        astronomy.

        Related constants:

        AERO,
        ASTRO

        The discipline is indicated by bit ?? in the index word of
        every data block of level 0 AC1, AC2 and AOS files. It may be
        retrieved using the following statement:
```

```
      \begin{verbatim}
      scan->Discpline = (index & 0x0000) ? ASTRO : AERO;
      \end{verbatim}
  */
  short   Discipline;

  /** Astronomy topic.
      In astronomy, the topical team which requested these observations.
      Not used by aeronomy.

      Related constants:

      SOLSYS,
      STARS,
      EXTGAL,
      LMC,
      PRIMOL,
      SPECTR,
      CHEM,
      GPLANE,
      GCENTR,
      GMC,
      SFORM,
      DCLOUD,
      SHOCKS,
      PDR,
      HILAT,
      ABSORB,
      ORION,
      CALOBS,
      COMMIS

      The topical team is extracted from the unvalidated time line (block 1)
      of an attitude file. It will be stored (with 6 characters)
      in comment fields enclosed in square brackets on the same line as
      the {\tt newompb} keyword. Example:
      \begin{verbatim}
      newompb:    0  [CALOBS W3(OH)                   -45.0  POS         ]
                      ------
      \end{verbatim}
  */
  short   Topic;

  /** Spectrum number in this orbit.
      An integer counting the individual spectra of one orbit,
      starting at 1. Because it is planned to write data to files grouped
      by orbits, this member will normally be counting spectra within a
      level 1b data file, as well. For averaged spectra it is the
      corresponding number of the first spectrum used for the average.
  */
  short   Spectrum;

  /** Observing mode.
      The observing mode in use when this spectrum was taken.

      Related constants:

```

```
        TPW,
        SSW,
        LSW,
        FSW
    */
    short   ObsMode;

    /** Type of spectrum.
        The type of the current spectrum.

        Related constants:

        SIG,
        REF,
        CAL,
        CMB,
        DRK,
        SK1,
        SK2,
        SPE,
        SSB,
        AVE

        Except for the last two qualifiers (which will be set by the level 1a
        to level 1b data reduction software) this information is retrieved
        mainly from the level 0 files of all spectrometers. All spectrometers
        store the position of the chopper wheel, i.e. SIG or REF. A REF
        spectrum can be qualified as CAL, SK1 or SK2 once the position of the
        calibration mirror is known. This is retrieved either from the
        spectrometer files (in the case of the AOS) or from {\tt STW.FBA}
        files, which record the position of this mirror with 1 second sampling.
    */
    short   Type;

    /** Frontend used.
        The frontend used for this observation.

        Related constants:

        REC_495,
        REC_549,
        REC_555,
        REC_572,
        REC_119,
        REC_SPLIT

        The information is retrieved from the input channel reported by
        each spectrometer. For the split modes and the currently accepted
        frontend configurations, AC1 will hold REC_495 data in its lower half
        and REC_549 in its upper half. AC2 will hold REC_572 data in its
        lower half and REC_555 in its upper half. During the data processing
        from level 1a to level 1b the two halfs of any split mode will be
        broken up in two individual spectra and stored as such.
    */
    short   Frontend;
```

```
/** Backend used.
    The backend used for this observation.

    Related constants:

    AC1,
    AC2,
    AOS
*/
short  Backend;

/** Indicates beam(s) on avoidance zones.
    The 16 bits of this word indicate possible hits of major
    sources of submillimetre emission by one of the skybeams or the
    main beam. E.g to test if the main beam was pointing at the Moon
    when a spectrum was taken during a limb scan measurement in aeronomy
    mode:
    \begin{verbatim}
    if (spectrum->SkyBeamHit &amp; MOONMB) {
       // oops! we were looking at the Moon
       ...
    }
    \end{verbatim}

    Related constants:

    SKYBEAMS,
    EARTH1,
    MOON1,
    GALAX1,
    SUN1,
    EARTH2,
    MOON2,
    GALAX2,
    SUN2,
    EARTHMB,
    MOONMB,
    JUPITERMB,
    SATURNMB

    The information is calculated based on the satellite's attitude
    and a low precision ephemeris for the Sun and Moon. The zone of
    avoidance for the galactic plane is based on the map of integrated
    CO 1-0 emission from the Columbia survey.
*/
short  SkyBeamHit;

/** Right ascension.
    The right ascension (J2000) of the source (direction of pointing)
    in degrees. Calculated from the quaternions stored in columns
    11-14 of block 3 of an attitude file.
 */
float  RA2000;

/** Declination.
    The declination (J2000) of the source (direction of pointing)
    in degrees. Calculated from the quaternions stored in columns
```

```
        11-14 of block 3 of an attitude file.
     */
    float   Dec2000;

    /** Source velocity.
        For astronomy, the velocity of source with respect to the local
        standard of rest (LSR) in m/s. It is extracted from the
        unvalidated time line (block 1) of an attitude file. The source
        velocity will be stored (with a {\tt %7.1f} format)
        in comment fields enclosed in square brackets on the same line as
        the {\tt newompb} keyword. Example:
        \begin{verbatim}
        newompb:    0  [CALOBS W3(OH)                   -45.0  POS           ]
                                                        -------
        \end{verbatim}
        For aeronomy, the relative velocity in m/s between the satellite
        and the tangent point. This is calulated from the satellites GPS
        velocity stored in columns 21-23 of block 3 of an attitude file.
    */
    float   VSource;

    /** Tangent point or map position.
        The union holds two structures, each consisting of three
        {\tt float} values with different meanings in astronomy and aeronomy.

        In astronomy the union holds a
        structure {\tt map}, describing map parameters in case the current
        spectrum is part of a mapping observation:
        \begin{verbatim}
        float Xoff // the map offset in the x-direction in degrees.
        float Yoff // the map offset in the x-direction in degrees.
        float Tilt // the position angle of the map in degrees.
        \end{verbatim}

        In aeronomy the union holds a structure {\tt tp}, describing the
        current tangent point:
        \begin{verbatim}
        float Longitude // geographical longitude of the tangent point in degrees.
        float Latitude  // geographical latitude of the tangent point in degrees.
        float Altitude  // geographical altitude of the tangent point in meter.
        \end{verbatim}

        Code to access the various members would look like this:
        \begin{verbatim}
        scan->u.tp.Longitude
        scan->u.map.Yoff
        \end{verbatim}

        The information on the map position is stored in block 1 of an
        attitude file. The map position will be stored
        in comment fields enclosed in square brackets on the same line as
        the {\tt newompb} keyword. Example:
        \begin{verbatim}
        newompb:    0  [CALOBS W3(OH)                   -45.0 MAP      60   -120   180.0]
                                                                     ----- -----  ------
        \end{verbatim}
        The information on the tangent point is stored in columns 27-29 of
```

```
        block 3 of an attitude file.
    */
    point u;

    /** Reference attitude.
        The reference attitude given as a quaternion (4-vector) as listed in
        columns 8-10 of block 3 of an attitude file.
    */
    double Qtarget[4];

    /** Achieved attitude.
        The achieved attitude given as a quaternion (4-vector) as listed in
        columns 11-14 of block 3 of an attitude file.
    */
    double Qachieved[4];

    /** Attitude error.
        The attitude error in degrees, describing the pointing uncertainty
        around the 3 principle axes of the satellite.
        Listed in columns 15-17 of block 3 of an attitude file.
    */
    double Qerror[3];

    /** Geocentric position.
        The geocentric position $X$,$Y$,$Z$ in meter of the satellite at the
        end of the observation as reported by the GPS receiver onboard
        of Odin and as listed in columns 18-20 of block 3 of an attitude file.
    */
    double GPSpos[3];

    /** Geocentric velocity.
        The geocentric velocity $\dot X$, $\dot Y$, $\dot Z$ in meter per
        second of the satellite at the end of the observation as reported
        by the GPS receiver onboard of Odin and as listed in columns 21-23
        of block 3 of an attitude file.
     */
    double GPSvel[3];

    /** Geocentric position of Sun.
        The geocentric position of the Sun in meter. Calculated based on
        current time using a low precision ephemeris. Accuracy is guaranteed
        to be one arcmin or better.
     */
    double SunPos[3];

    /** Geocentric position of Moon.
        The geocentric position of the Moon in meter. Calculated based on
        current time using a low precision ephemeris. Accuracy is guaranteed
        to be one arcmin or better.
    */
    double MoonPos[3];

    /** Solar zenith angle.
        The solar zenith angle in degrees. Calculated from the position of
        the Sun and therefore with the same intrinsic accuracy.
     */
    float  SunZD;
```

```

/** Velocity with respect to the Earth.
    The velocity of the satellite with respect to the Earth in
    meter per second. Calculated from current time and satellite
    GPS position and velocity.
*/
float  Vgeo;

/** Velocity with respect to LSR.
    In astronomy, the velocity of the satellite with respect to the
    local standard of rest (LSR) in meter per second. Calculated from
    current time and satellite GPS position and velocity.
    Not used by aeronomy.
*/
float  Vlsr;

/** Calbration temperature.
    The Rayleigh Jeans temperature in Kelvin of the calibration load
    used during the intensity calibration of this spectrum. The physical
    temperature of the load is retrieved from level 0 house keeping
    data files.
*/
float  Tcal;

/** System temperature.
    The mean value of the system temperature in Kelvin
    used during the intensity calibration of this spectrum.
    Calculated during calibration.
*/
float  Tsys;

/** Diplexer path length.
    The path length in meter of the SSB diplexer. Extracted from house
    keeping data files.
*/
float  SBpath;

/** Local oscillator frequency.
    The (first) local oscillator frequency in Hz used for this observation
    in the rest frame of the satellite. Calculated from the information
    on HRO and PRO frequency stored in level 0 house keeping data files.
*/
double LOFreq;

/** Sky frequency.
    The frequency in Hz of the centre channel in the rest frame of the
    satellite, i.e not Doppler corrected. Calculated from the LO frequency
    and the value of the IF frequency, which will be either +3900 MHz or
    -3900 MHz. The sign is calculated based on the setting of the SSB
    tuning mechanism, stored in the level 0 house keeping data files.

    Note that the index of the centre channel {\tt cc} is defined as
    \begin{verbatim}
    cc = spectrum->Channels/2;
    \end{verbatim}
    with all the implications of an integer division. This means, that
    for an odd number of channels, the division yields the correct array
```

```
        index of the centre channel, for an even number of channels we get
        the index of the channel just above the centre of the band (which falls
        between two channels).
    */
    double SkyFreq;

    /** Rest frequency.
        The frequency in Hz of the centre channel in the rest frame of the
        observed object, i.e Doppler corrected. The same definition of
        centre channel applies as for the previous member. Calculated from
        sky frequency and applicable Doppler correction (see Vlsr).

        For astronomy, the LSR velocity of the source supplied via block 1
        of an attitude file together with the calculated LSR velocity of
        the satellite is used to translate the sky frequency to a rest frame
        centred on the source.

        For aeronomy, the relative velocity of the satellite in the direction
        of the tangent point is used to translate the sky frequency to an
        earth fixed reference frame.

        The frequency correction is calculated using the radio convention:
        \begin{verbatim}
        SkyFreq = RestFreq*(1 - v/c)
        \end{verbatim}
    */
    double RestFreq;

    /** Frequency of max.suppression.
        The sideband frequency in Hz corresponding to the frequency of
        maximum suppression in the rest frame of the satellite,
        i.e not Doppler corrected.
    */
    double MaxSuppression;

    /** Soda version.
        The version number of the attitude reconstruction software (SODA)
        used at SSC during production of attitiude files.
     */
    double SodaVersion;

    /** Frequency resolution.
        The spacing in Hz between neighbouring channels for this spectrum.
    */
    double FreqRes;

    /** Frequency calibration coefficients.
        Frequency calibration coefficients in Hz for the spectrometer in
        use.

        For the correlators, these are the four local oscillator frequencies
        of the SSB modules, in the order in which individual bands are read
        out from the instrument, i.e. 1,3,2,4 when compared to Omnisys
        documentation.
        Typical values:
        \begin{verbatim}
        FreqCal[0] =  3600.0e+06;
```

```
        FreqCal[1] =  3800.0e+06;
        FreqCal[2] =  4000.0e+06;
        FreqCal[3] =  4200.0e+06;
        \end{verbatim}

        For the acousto optic spectrometer, these are the fit results from the
        last frequency comb measurement performed.
        Typical values:
        \begin{verbatim}
        FreqCal[0] =  2.100e+09;
        FreqCal[1] =  6.200e+05;
        FreqCal[2] =  4.000e+00;
        FreqCal[3] = -1.000e-02;
        \end{verbatim}
    */
    double FreqCal[4];

    /** Backend mode.
        The integration mode of the backend during the observation.

        Related constants

        AC_XHIRES,
        AC_HIRES,
        AC_MEDRES,
        AC_LOWRES,
        AC_YHIRES,
        AC_SPLIT,
        AC_UPPER,
        ADC_SEQ,
        AOS_LONG,
        AOS_SHORT,
        AOS_HALF,
        AOS_FOUR,
        AOS_CENTRE,
        AOS_WINGS,
        AOS_WINDOW

        The information is retrieved from values reported by the
        spectrometers.
    */
    int     IntMode;

    /** Integration time.
        The integration time in seconds, i.e. the duration of this
        observation.
     */
    float   IntTime;

    /** Effective integration time.
        The effective integration time in seconds, i.e. you will get the
        noise level in this spectrum by using this time and the system
        temperature from above and the usual radiometer formula:
        \begin{verbatim}
        dT = Tsys/sqrt(df * EffTime)
        \end{verbatim}
        where {\tt df} is FreqRes from above.
```

```
   */
   float   EffTime;

   /** Number of channels.
       The number of channels in this spectrum. The maximum number
       of channels that may occur is given by the number of channels in
       the AOS, see below.
   */
   int      Channels;

   /** Channel data.
       The array holding the actual spectrum. To loop through the
       array use:
       \begin{verbatim}
       int channel;

       for (channel = 0; channel < spectrum->Channels; channel++) {
           // process each channel
           spectrum->data[channel] = ...;
       }
       \end{verbatim}

       Related constant:

       MAXCHANNELS
   */
   float   data[MAXCHANNELS];
};
//@}

#endif
```