

spi_psd_adcgain

User Manual

Version 1.4.1
30 September 2002

Jürgen Knödseder
Centre d'Etude Spatiale des Rayonnements
knodseder@cesr.fr
<http://www.cesr.fr/~jurgen/index.html>

Note to the user

This software has been written to analyse data of the SPI telescope onboard INTEGRAL. Particular care has been taken in making the software user friendly and well documented. If you appreciated this effort, and if this software and User Manual were useful for your scientific work, the author would appreciate a corresponding acknowledgment in your published work.

Contents

1	Introduction	1
2	Getting started	1
3	Parameter file	3
4	Interface definition	6
5	Algorithm	7
6	Alerts	8
7	Error codes	8

1 Introduction

The executable `spi_psd_adcgain`, written in the ANSI C++ language, derives the PSD ADC gain correction factors from PSD curve data. PSD ADC gain correction factors are employed for onboard pulse shape analysis in order to correct for small differences between the gain and offset of the four pulse shape A/D convertors (recall that the PSD has four A/D convertors for pulse shape digitisation; each convertor is clocked at 25 MHz while they are delayed by 10 ns with respect to each other; in this manner a total sampling frequency of 100 MHz is reached). The PSD ADC gains and offsets have to be monitored regularly in order to detect drifts in the performance of the electronics. For this purpose, `spi_psd_adcgain` implements a limit violation alert logic that automatically generates ISDC alerts upon task execution. Eventually, this monitoring will lead to an update of the PSD ADC gain correction factors (these factors are set by telecommand).

`spi_psd_adcgain` has been designed to execute in any kind of pipeline, such as the science window pipeline or the revolution pipeline (however it could also be applied to an observation group for deep performance analysis). Since a sufficient number of PSD curves is required for reliable ADC gain correction, the **revolution pipeline is the privileged location for `spi_psd_adcgain`**. `spi_psd_adcgain` has been designed to append regularly ADC gain corrections to a file, and hence to build a time history of the ADC gains. To make the logic work, `spi_psd_adcgain` needs an index group on input, which it will search automatically for sufficient **new data** (i.e. data that are dated after the last entry in result file) for ADC gain correction computation. If not enough new data is available, `spi_psd_adcgain` will do nothing.

`spi_psd_adcgain` has been developed under ISDC support platform 4.1 and requires `spi_psdlib` version 1.5.0 and `spi_toolslib` version 1.8.0 or higher.

2 Getting started

Before installing `spi_psd_adcgain`, make sure that the ISDC support platform 4.1 or higher is installed on your system, and that you have installed the libraries `spi_psdlib` version 1.5.0 and `spi_toolslib` version 1.8.0 or higher.

After downloading the `spi_psd_adcgain.tar.gz` file, step into a directory that should hold the distribution, move the `spi_psd_adcgain.tar.gz` file into this directory and type:

```
$ gunzip spi_psd_adcgain.tar.gz
$ tar xvf spi_psd_adcgain.tar
```

The first command uncompresses the distribution file, the second unpacks the files.

Before configuration, the distribution needs to be reset to a clean state. To do this, type

```
$ make distclean
```

Then, configure the distribution. It is assumed here that you have previously installed the ISDC support platform, thus you should type

```
$ ~/bin/ac_stuff/configure
```

Finally, build the distribution by typing

```
$ make global_install
```

To perform a unit test, type

```
$ make test
```

Make sure that the test data `spi_test_data-1.0.tar.gz` are available at your site (they should reside in a directory whose name is defined by the `ISDC_TEST_DATA_DIR` environment variable).

3 Parameter file

```
#####
#
#           Centre d'Etude Spatiale des Rayonnements           #
#           (in collaboration with ISDC)                       #
#
#           SPI PSD ADC gain correction                         #
#
# -----#
#
# File:      spi_psd_adcgain.par                               #
# Version:   1.4.0                                           #
# Component: osm                                             #
#
# Author:    Juergen Knoedlseder                             #
#            knodlseder@cesr.fr                              #
#            CESR                                             #
#
# Purpose:   Parameter file of the SPI PSD gain correction   #
#            executable                                       #
#
# History:   1.4.0 20-Aug-2002 First ISDC delivery (Rev. 4)  #
#
#####
#
# Input DOLs
#=====
inDOL,      s, ql, "swg_prp.fits[1]",,,"Input Group DOL (SWG/OG/IDX)"
alertDOL,   s, ql, "psd_limits_idx.fits[1]",,,"Alert Limit DOL (File/IDX)"
#
# Output DOL
#=====
outDOL,     s, ql, "adc.fits",,,"Output DOL (HDU optionally)"
#
# OBT limits
#=====
minOBT,     s, ql,      "",,,"Curve usage minimum OBT"
maxOBT,     s, ql,      "",,,"Curve usage maximum OBT"
append,     b, ql,     yes,,,"Append minimum OBT to last results ?"
slice,      b, ql,     yes,,,"Split time interval in constant ONTIME intervals ?"
nopart,     b, ql,     yes,,,"Skip partial time intervals ?"
ontime,     r, ql,    7200.0,,,"Constant ONTIME slice (seconds)"
#
# ADC analysis parameters
#=====
dopmax,     i, h,      6,  1,  19,,,"Maximum degree of polynomial used for baseline fit"
fitmin,     r, h,    505.0,-0.5,785.0,,,"First timestep (ns) to use for baseline fit"
fitmax,     r, h,    785.0,-0.5,785.0,,,"Last timestep (ns) to use for baseline fit"
#
# SPI mode usage
#=====
usemode0,   b, h,     yes,,,"Use OPER mode ?"
```

```

usemode2, b, h, yes,,, "Use CALB mode ?"
usemode3, b, h, yes,,, "Use DIAG mode ?"
#
# PSD channel usage
#=====
usedete00, b, h, yes,,, "Use PSD channel 0 ?"
usedete01, b, h, yes,,, "Use PSD channel 1 ?"
usedete02, b, h, yes,,, "Use PSD channel 2 ?"
usedete03, b, h, yes,,, "Use PSD channel 3 ?"
usedete04, b, h, yes,,, "Use PSD channel 4 ?"
usedete05, b, h, yes,,, "Use PSD channel 5 ?"
usedete06, b, h, yes,,, "Use PSD channel 6 ?"
usedete07, b, h, yes,,, "Use PSD channel 7 ?"
usedete08, b, h, yes,,, "Use PSD channel 8 ?"
usedete09, b, h, yes,,, "Use PSD channel 9 ?"
usedete10, b, h, yes,,, "Use PSD channel 10 ?"
usedete11, b, h, yes,,, "Use PSD channel 11 ?"
usedete12, b, h, yes,,, "Use PSD channel 12 ?"
usedete13, b, h, yes,,, "Use PSD channel 13 ?"
usedete14, b, h, yes,,, "Use PSD channel 14 ?"
usedete15, b, h, yes,,, "Use PSD channel 15 ?"
usedete16, b, h, yes,,, "Use PSD channel 16 ?"
usedete17, b, h, yes,,, "Use PSD channel 17 ?"
usedete18, b, h, yes,,, "Use PSD channel 18 ?"
#
# Limit checking definitions
#=====
limcheck, b, h, yes,,, "Perform limit checking ?"
alert0, b, h, yes,,, "Generate level 0 alerts ?"
alert1, b, h, yes,,, "Generate level 1 alerts ?"
alert2, b, h, yes,,, "Generate level 2 alerts ?"
alert3, b, h, yes,,, "Generate level 3 alerts ?"
minCRVE, i, h, 500,,, "Minimum number of PSD curves for limit checking"
#
# ISDC Standard Parameters
#=====
clobber, b, h, no, , "Overwrite existing data structures ?"
mode, s, h, "ql",,, "Execution mode"

```

The following parameters have to be specified:

- **inDOL** specifies the input DOL (science window, observation group, or index file) for which the PSD ADC gain correction factors should be derived. An ISDC level of PRP is required with full OBTs.
- **alertDOL** (optional) if alert limit checking is requested (**limcheck = yes**), this parameter specifies the DOL of the alert limit file [SPI.-ALRT-LIM] or the alert limit index [SPI.-ALRT-LIM-IDX] (including the HDU).
- **outDOL** specifies the output **filename** into which the PSD ADC gain correction factors will be written. The specification of the HDU [SPI.-ADC.-PSD] is optional and is not required by the task. If the data structure exists already, **spi_psd_adcgain** appends rows to the existing table. If the data structure or the file does not exist, **spi_psd_adcgain** creates a new file/HDU.
- **minOBT** specifies the minimum OBT limit of the curves that should be used for PSD ADC gain correction determination. The OBT format is a character string. Leading 0 may be omitted. If the

character string is empty, or if any non-number character is specified (such as "no" for example), no minimum OBT limit is applied (and data accumulation starts with the first curve in the input group).

- **maxOBT** specifies the maximum OBT limit of the curves that should be used for PSD ADC gain correction determination. The OBT format is a character string. Leading 0 may be omitted. If the character string is empty, or if any non-number character is specified (such as "no" for example), no maximum OBT limit is applied (and data accumulation stops with the last curve in the input group).
- **append** specifies if the minimum OBT limit should be set to the last OBT that occurs in the output file (specified by **outDOL**) in order to produce a continuous set of PSD ADC gain corrections. The last OBT will be extracted from the keyword **OBTLAST** in the output file. **This parameter is only active if minOBT has not been set by the user**, i.e. **minOBT** has precedence and will not be overwritten.
- **slice** specifies if the input group should be "sliced" into time frames of constant **ONTIME** (the **ONTIME** is the time, specified in seconds, during which SPI science data were accumulated and made available to the observer). Such time slices are also referred to as Good Time Intervals (GTIs).
- **nopart** (optional) if **slice = yes**, specifies if partial time slices, i.e. time slices with durations that are shorter than the requested **ONTIME**, should be skipped. Partial time slices may occur at the end of a data stream, and to assure a uniform quality of the PSD ADC gain corrections it is recommended to set this parameter to **yes**. Together with **append = yes**, re-execution of **spi-psd_adcgain** at a later time will append new time slices that start with the OBT of the last appended time slice.
- **ontime** (optional) if **slice = yes**, specifies the **ONTIME** duration in seconds of each time slice, also referred to as Good Time. A reasonable **ONTIME** is of the order of several hours. Note that if **nopart = no**, the last time slice has generally an effective **ONTIME** that is shorter than the specified value, since in general, the available **ONTIME** is not an integer multiple of the value specified by **ontime**.
- **dopmax** specifies the maximum degree of the polynomial that will be used to estimate the PSD ADC gain correction factors. The minimum setting is 1 (corresponding to a linear function), the maximum is 19. Values above 6 or 7 have shown to produce invalid fit results, yet this does not disturb the executable since always the best degree of polynomial will be taken. As default, a value of 6 should be specified.
- **fitmin** specifies together with **fitmax** a time interval of the PSD pulse shapes (or curves) that should be used for ADC gain correction estimation. Usually, this time interval is the pulse baseline, which normally is found between 500 and 790 ns. By default, 505 ns is used (since the PSD time step is 10 ns, 505 ns specifies a time bin that starts at 500 ns). The unit of this parameters is ns.
- **fitmax** specifies together with **fitmin** a time interval of the PSD pulse shapes (or curves) that should be used for ADC gain correction estimation. Usually, this time interval is the pulse baseline, which normally is found between 500 and 790 ns. By default, 785 ns is used (since the PSD time step is 10 ns, 785 ns specifies a time bin that ends at 790 ns). The unit of this parameters is ns.
- **usemoden** specifies which science modes should be used for PSD ADC gain correction estimation, where $n = 0, 2, 3$ for the three SPI science modes **OPER**, **CALB**, and **DIAG** that produce PSD curve data (the **EMER** mode has been excluded since it does not produce PSD curve data). If all parameters are set to **yes** (the default), all curves that can be found in the data will be used for ADC gain correction estimation. However, a new time interval will be inserted in the **SPI.-ADC.-PSD** HDU for each science mode change.
- **usedetenn** specifies which detectors should be used for PSD ADC gain correction estimation, where $nn = 00 - 18$. By default, all detectors will be used (**yes**), and pulses will be averaged from all detectors for ADC gain correction determination. If, however, one or several PSD channels are too noisy, they may be excluded from the analysis by setting the correspondent field to **no**.

- **limcheck** specifies if alert limit checking should be performed by **spi_psd_adcgain**. If set to **yes**, **spi_psd_adcgain** compares all ADC gain corrections to the limits that are specified in the alert limit file (see **alertDOL**) and (optionally) creates ISDC alerts (see parameters **alert0** to **alert3**). Alert limit checking will be only performed for time slices that show sufficient curve statistics. The curve statistics limit is defined by the parameter **minCRVE**.
- **alert0** (optional) if alert limit checking is enabled (**limcheck = yes**), generates level 0 ISDC alerts.
- **alert1** (optional) if alert limit checking is enabled (**limcheck = yes**), generates level 1 ISDC alerts.
- **alert2** (optional) if alert limit checking is enabled (**limcheck = yes**), generates level 2 ISDC alerts.
- **alert3** (optional) if alert limit checking is enabled (**limcheck = yes**), generates level 3 ISDC alerts.
- **minCRVE** (optional) if alert limit checking is enabled (**limcheck = yes**), specifies the minimum required number of PSD curves (CRVE) to initiate alert limit checking. This parameter avoids alert limit checking in case of insufficient curve statistics. To achieve reasonably good ADC corrections, a minimum number of curves of the order of ~ 1000 should be requested (for a nominal curve rate of one each four seconds, this implies an average acquisition duration of about one hours and ten minutes, hence **ONTIME** > 4000 in this case).
- **clobber** ISDC standard parameter (not used so far).
- **mode** ISDC standard parameter (not used so far).

4 Interface definition

On input, **spi_psd_adcgain** reads PSD curves from either a science window group or an observation group. The analysis level **PRP** is required with OBT information for each PSD curve. Also PSD onboard analysis errors are gathered from the **PRP** data, and only pulse shapes will be used that had no onboard error (pulse shapes with onboard error may have an invalid baseline, hence they would falsify the ADC gain determination).

On output, **spi_psd_adcgain** adds rows to the **SPI.-ADC.-PSD** data structure with gain correction factors for the analysed PSD curves. The number of rows that are added depends on the number of time intervals that have been found by **spi_psd_adcgain**. If no time slicing has been specified (**slice = no**) and only a single SPI science mode is present in the input group, only one row will be added. If time slicing has been selected, the number of rows will depend on the number of time slices that have been found. Also, each new SPI science mode that is found in the input group will add a new row to the data structure. Optionally, time slices that are shorter than 95% of the requested **ONTIME** may be skipped, thus assuring a uniform quality (in terms of data statistics) of the PSD ADC gain correction results.

spi_psd_adcgain fills all columns of the **SPI.-ADC.-PSD** HDU, hence it can be considered as complete after the task has finished. The **OBTFIRST** and **OBTLAST** keywords are also updated, so that index group generating tools may be used to assess the validity interval of the data structure. In particular, **spi_psd_adcgain** may access the **OBTLAST** keyword set by a previous run if continuous time slices should be added to the output file (parameter **append = yes**). The following columns are filled by **spi_psd_adcgain**:

- **OBT_START** : OBT start of the time interval of the actual row.
- **OBT_STOP** : OBT stop (or end) of the time interval of the actual row.
- **ONTIME** : ontime for this row in seconds.
- **PSD_ADC_GAIN** : gain correction factors for all four A/D convertors.
- **PSD_ADC_OFFSET** : offset correction factors for all four A/D convertors.

- **CRVE_NUM** : number of PSD curves used to determine the gain correction factors.

Optionally, `spi_psd_adcgain` is able to generate ISDC alerts if some of the ADC gain corrections fall out of the defined limits. The alert limits are defined by an `SPI.-ALRT-LIM` structure from which the following columns are used by `spi_psd_adcgain` for alert generation:

- **PAR_NAME** specifies the parameter for which the limits apply. Valid parameter names are `PSD_ADC_GAIN` and `PSD_ADC_OFFSET`. If one of those parameter names is defined, the specified limits apply to **all** 4 A/D convertors. By adding `_Ln` to the parameter name (where n runs from 0 to 3), ADC gain correction limits may be specified for a given convertor (for example `PSD_ADC_OFFSET_L3` specifies the limits for the offset of the A/D convertor 3). Convertor specific parameters have precedence over common parameters (i.e. those without the `_Ln` extension), hence one may define common limits for all 4 A/D convertors and overwrite a few limits for specific convertors by specifying explicitly the convertor number.
- **MIN_VAL** specifies the lower ADC gain correction limits (inclusive) for the four ISDC alert levels (DAL table columns 1-3 corresponds to alert levels 0-3).
- **MAX_VAL** specifies the upper ADC gain correction limits (inclusive) for the four ISDC alert levels (DAL table columns 1-3 corresponds to alert levels 0-3).
- **SUB_ASSEMBLY** specifies the SPI PSD sub-assembly and must contain the entry `SPI_PSD`.

All other columns (`OBT_START`, `OBT_END`, `CHECK_MODE`, `ALERT_DELAY`) of the `SPI.-ALRT-LIM` structure are ignored. The validity of the alert limit file is defined by the two keywords `VSTART` and `VSTOP`. If an alert limit index is used, these keywords are used to select the alert limit file that is appropriate for the ADC gain correction validity time interval. If the ADC gain correction validity time interval stops before the validity of the earliest alert limits, the earliest alert limits are used by `spi_psd_adcgain` (a warning is issued by `spi_psd_adcgain` in this case). If the validity time interval starts after the validity of the last alert limits, the last alert limits are used by `spi_psd_adcgain` (a warning is issued by `spi_psd_adcgain` in this case). If the ADC gain correction validity time interval overlaps with the transition of two (or more) alert limits files, those alert limits are applied that have the longest time overlap with the ADC gain correction validity time interval.

5 Algorithm

The PSD ADC gain correction factors are derived from the baseline of the PSD pulses. The baseline is defined as the last (say 25) bins of the PSD pulse shape which follow the detector pulse. Normally, this baseline is rather flat and represents the baseline current of the front-end electronics.

Since the four A/D convertors that digitise the detector current pulses are common for all 19 detection channels, all PSD curves are used irrespectively of their detector identifier. Thus, in a first step, an average PSD curve is derived by summing all available PSD curves in the input data structure. Curves for which an on-board analysis error occurred are removed (it is likely that for these curves the baseline is badly defined). Further curve filters are possible, but not implemented so far.

The baseline, which is defined by the two task parameters `fitmin` and `fitmax` is then fitted by a polynomial function in order to get an adequate average description of the baseline shape. The degree of the polynomial varies between 1 (i.e. a linear function) and the value given by the task parameter `dopmax` (typically 6). Fits are done for all possible degrees between these limits. The optimum fit is then given by the polynomial that produces the smallest reduced χ^2 , i.e. where $\chi^2/d.o.f$ takes a minimum (where *d.o.f.* is the degrees of freedom of the fit). Using this method, the optimum polynomial is the polynomial that describes the baseline sufficiently well without introducing unnecessary parameters (that's a kind of Ockham's razor approach).

In the actual version of this program, no gain correction is assumed (i.e. $gain_i = 1.0$) and only ADC offset corrections $offset_i$ are calculated. Following the polynomial fit, the offset for ADC i is then calculated using

$$offset_i = \frac{\sum_{k=fitmin/4}^{fitmax/4} curve_{4k+i}}{\sum_{k=fitmin/4}^{fitmax/4}}. \quad (1)$$

Optional alert limit checks are performed with higher alert levels preceding lower alert levels, i.e. an alert of the highest possible level is generated. Minimum parameter limits are checked before maximum parameter limits, and minimum limits have precedence over maximum limits (this is not really of relevance since a parameter can not both violate the minimum and maximum limit **unless the alert limit file has not been set up correctly**, i.e. the minimum limit is always smaller or equal to the maximum limit). Alert limits are inclusive, i.e. a minimum limit violation alert is generated if

$$PARAMETER < MIN_VAL \quad (2)$$

is fulfilled, and a maximum limit violation alert is generated if

$$PARAMETER > MAX_VAL \quad (3)$$

is fulfilled.

Limit violation alerts are only generated if

$$CRVE_NUM \geq minCRVE \quad (4)$$

is fulfilled, where `CRVE_NUM` is the total number of curves in a time slice, including also curves for which a PSD algorithm error occurred (see the `spi_psdlib` User Manual for a definition of PSD algorithm errors).

6 Alerts

`spi_psd_adcgain` may optionally generate alerts that signal possible PSD/SPI misfunctions. The following list provides the alert parameters and the actions that should be taken in case of occurrence of the alerts. The alert parameter in the alert message is followed by the extension `_Ln` where $n=0-3$ specifies the ADC convertor number for which the alert occurred.

If the action **standard** is specified in the table, the standard alert action should be performed (**TBD**).

Parameter	Level	Action
PSD_ADC_GAIN	0-3	standard
PSD_ADC_OFFSET	0-3	standard

7 Error codes

The executable `spi_psd_adcgain` may stop with the following error codes:

<code>SPI_PSD_ADCGAIN_ERROR_MEM_ALLOC</code>	-230000
<code>SPI_PSD_ADCGAIN_ERROR_INDEX_SELECT</code>	-230001

They have the following meaning:

- `SPI_PSD_ADCGAIN_ERROR_MEM_ALLOC` : the allocation of dynamical memory has failed. Probable your system resources are too limited to run this task. If you cannot increase your resources you may reduced the number `SPI_PSD_ADCGAIN_BUFFER` in the `spi_psd_adcgain.h` header file and recompile the code.

- **SPI_PSD_ADCGAIN_ERROR_INDEX_SELECT** : while searching a single member in an index group, DAL3GEN returned more than one member. This should never happen. If this error occurs, it is likely that the index you specified on input is corrupted.

In addition, all errors that may occur in calls to ISDC support functions (such as for example DAL, RIL or PIL) are forwarded. Please consult the ISDC web pages for getting information about these error codes.