

spi_psd_performance

User Manual

Version 1.5.0
21 August 2002

Jürgen Knödseder
Centre d'Etude Spatiale des Rayonnements
knödseder@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	5
5	Algorithm	9
6	Alerts	11
7	Error codes	12

1 Introduction

The executable `spi_psd_performance` derives **PSD performance indicators** from PSD curve and event data. The performance data are derived for each of the 19 PSD channels separately to allow for channel-to-channel variations. These performance data should be monitored throughout the entire INTEGRAL mission at ISDC, and their values should be checked against limit violations. For this purpose, `spi_psd_performance` implements a limit violation alert logic that automatically generates ISDC alerts upon task execution.

Most of the performance indicators are derived from the regularly received PSD curves, and are averages or random mean scatters (rms) around the averages of PSD curve characteristics (such as curve starttime, endtime, duration, baseline, baseline slope, noise, fit quality). In addition, the fraction of PSD analysis errors and onground pulse fitting errors is also reported.

`spi_psd_performance` 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 events and PSD curves is required for a reliable determination of the PSD performance, the **revolution pipeline is the privileged location for `spi_psd_performance`**. `spi_psd_performance` has been designed to append regularly PSD performance indicators to a file, and hence to build a time history of the PSD performance. To make the logic work, `spi_psd_performance` 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 PSD performance evaluation. If not enough new data is available, `spi_psd_performance` will do nothing.

`spi_psd_performance` is written in the ANSI C++ language. The task 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_performance`, 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_performance.tar.gz` file, step into a directory that should hold the distribution, move the `spi_psd_performance.tar.gz` file into this directory and type:

```
$ gunzip spi_psd_performance.tar.gz
$ tar xvf spi_psd_performance.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 performance indicators derivation           #
#
# -----#
#
# File:      spi_psd_performance.par                           #
# Version:   1.5.0                                           #
# Component: osm                                             #
#
# Author:    Juergen Knoedlseder                             #
#            knodlseder@cesr.fr                             #
#            CESR                                           #
#
# Purpose:   Parameter file of the SPI PSD performance indicators #
#            derivation executable                           #
#
# History:   1.5.0 21-Aug-2002 First ISDC release (Rev. 5)  #
#
#####
#
# Input DOLs
#=====
inDOL,      s, ql, "swg_prp_idx.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, "performance.fits",,,,"Output DOL (HDU optionally)"
#
# OBT limits
#=====
minOBT,     s, ql,      "",,,,"Event usage minimum OBT"
maxOBT,     s, ql,      "",,,,"Event 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,    3600.0,,,"Constant ONTIME slice (seconds)"
#
# 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 ?"
minPE,      i, h,    1000,,,"Minimum number of PE for limit checking"
minCRVE,    i, h,     100,,,"Minimum number of CRVE 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 group, observation group, or index group) for which the PSD performance indicators should be derived. If the ISDC level of the input group is PRP (which is the minimum required level), the **POST_ERR_FRACT** column of the **SPI.-PERF-PSD** HDU will not be filled, but is defaulted to 0.0. If the ISDC level of the input group is COR, the **POST_ERR_FRACT** column is filled.
- **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 performance information is written. The specification of the HDU [**SPI.-PERF-PSD**] is optional, but not required by the task. If the data structure exists already, **spi_psd_performance** appends rows to the existing table. If the data structure or the file does not exist, **spi_psd_performance** creates a new file/HDU.
- **minOBT** specifies the minimum OBT limit of the events that should be used for performance 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 event in the input group).
- **maxOBT** specifies the maximum OBT limit of the events that should be used for performance 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 event 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 performance indicators. 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).
- **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 performance indicators it is recommended to set this parameter to **yes**. Together with **append = yes**, re-execution of **spi_psd_performance** 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 of each time slice. Note that 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**.
- **limcheck** specifies if alert limit checking should be performed by **spi_psd_performance**. If set to **yes**, **spi_psd_performance** compares all performance parameters 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 detectors and time intervals that show sufficient event statistics. The event statistics limits are defined by the parameters **minPE** and **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.
- **minPE** (optional) if alert limit checking is enabled (**limcheck = yes**), specifies the minimum required number of PSD events (PE) for one PSD detection channel to initiate alert limit checking. This parameter avoids alert limit checking in case of insufficient event statistics. Typically, since percent levels should be tested, this parameter should be ≥ 1000 (**TBC**).
- **minCRVE** (optional) if alert limit checking is enabled (**limcheck = yes**), specifies the minimum required number of PSD curves (CRVE) for one PSD detection channel to initiate alert limit checking. This parameter avoids alert limit checking in case of insufficient curve statistics. To achieve a reasonably good average, a minimum of the order of ~ 100 (**TBC**) curves should be requested (for a nominal curve rate of one each four seconds, this implies an average acquisition duration of about two hours).
- **clobber** ISDC standard parameter (not used so far).
- **mode** ISDC standard parameter (not used so far).

4 Interface definition

spi_psd_performance derives PSD performance indicators from the PSD curves and events that are present in the specified input group (which can be either a science window group, an observation group, or an index group). **spi_psd_performance** needs an ISDC level of **COR** with the **PSD_CORFLAG** column of PSD events (**PE**) filled to build a complete result data structure (hence **spi_psd_postprocess** has to be called before **spi_psd_performance**). However, **spi_psd_performance** may also be applied to data of **PRP** level only, which however will result in invalid values in the **POST_ERR_FRACT** column of the result data structure (in this case the values of the column will be set to zero).

An **OBT** interval may be specified that selects a particular sub-interval for which the performance parameters should be determined. In addition, one may slice the specified **OBT** interval in time intervals of constant **ONTIME** (this option guarantees a constant integration time for all time slices, hence comparable uncertainties in all quantities as function of time).

For each time-interval, **spi_psd_performance** adds one row to the **SPI.-PERF-PSD** result data structure. **spi_psd_performance** fills all columns of the **SPI.-PERF-PSD** data structure, 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_performance** 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_performance**:

- **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.
- **AVG_STARTTIME** : average pulse start time in units of 10 ns.

- **AVG_ENDTIME** : average pulse end time in units of 10 ns.
- **AVG_DURATION** : average pulse duration in units of 10 ns.
- **AVG_BASELINE** : average pulse baseline in units of A/D converter steps (digits).
- **AVG_BASE_SLOPE** : average pulse baseline slope in units of A/D converter steps per ns (digits/ns).
- **AVG_NOISE** : average pulse noise in units of A/D converter steps (digits).
- **AVG_CHISQR** : average χ^2 of the onground template library fitting. For χ^2 calculation, the baseline noise has been applied (see **spi_psdlib** User Manual).
- **RMS_STARTTIME** : rms of the pulse start time in units of 10 ns.
- **RMS_ENDTIME** : rms of the pulse end time in units of 10 ns.
- **RMS_DURATION** : rms of the pulse duration in units of 10 ns.
- **RMS_BASELINE** : rms of the pulse baseline in units of A/D converter steps (digits).
- **RMS_BASE_SLOPE** : rms of the pulse baseline slope in units of A/D converter steps per ns (digits/ns).
- **RMS_NOISE** : rms of the pulse noise in units of A/D converter steps (digits).
- **RMS_CHISQR** : rms of the χ^2 of the onground template library fitting.
- **PSD_ERR_FRACT** : fraction of events with pulse shape analysis onboard errors. Pulse shape analysis onboard errors may occur if either the pulse is not appropriate (pulse too short or saturated, pulse pileup, bad pulse baseline, etc.) or if no valid template library is available for pulse fitting (which means that a PSD configuration error occurred). The following 19 columns provide a breakdown of the various error types that may occur.
- **PSD_ERR_INVALID** : fraction of events without a valid template library loaded into the PSD memory. This signals possible PSD configuration errors.
- **PSD_ERR_NO_CORR** : fraction of events with failed DFEE - PSD information correlation. If this fraction becomes too high a **potential problem between the PSD - DFEE - DPE data links has occurred**.
- **PSD_ERR_NO_CONF** : fraction of events for which no PSD configuration information has been found for PSD science word decoding. Such errors may occur if the task **dp_spi_psd** (which runs in the science window pipeline) was unable to obtain the PSD configuration from a previous science window, either because the **PREVSWID** keyword in the science window group has been not set correctly, or because the previous science window was missing at the time when **dp_spi_psd** was executed (due to a pre-processing problem).
- **PSD_ERR_NO_INX2TTP** : fraction of events for which the conversion from template index to time-to-peak value failed. This error is probably due to an invalid library specified for the task **dp_spi_psd**.
- **PSD_ERR_BAD_DETE** : fraction of events with invalid onboard detector identifier. Such errors should never occur. If they occur they **may indicate a hardware problem in the PSD sub-assembly**.
- **PSD_ERR_BAD_DECODE** : fraction of events that showed a PSD science word decoding error. Such errors should never occur.
- **PSD_ERR_SAT** : fraction of saturated events. This fraction depends on the spectral shape of the instrumental background, and if the background remains stable, should not vary tremendously.
- **PSD_ERR_PA_ZERO** : fraction of events with zero baseline subtracted pulse area. These events can not be fitted correctly by the onboard fitting algorithm.

- `PSD_ERR_PA_SMALL` : fraction of events with pulse area below the specified lower threshold (specified in the algorithm parameter block).
- `PSD_ERR_PA_LARGE` : fraction of events with pulse area above the specified upper threshold (specified in the algorithm parameter block).
- `PSD_ERR_ATTP_EARLY` : fraction of events with a too early absolute time-to-peak value.
- `PSD_ERR_ATTP_LATE` : fraction of events with a too late absolute time-to-peak value.
- `PSD_ERR_BASE_LOW` : fraction of events with a baseline below the specified lower threshold (specified in the algorithm parameter block).
- `PSD_ERR_BASE_HIGH` : fraction of events with a baseline above the specified upper threshold (specified in the algorithm parameter block).
- `PSD_ERR_BASE_START` : fraction of events where the pulse starts in the front baseline.
- `PSD_ERR_BASE_END` : fraction of events where the pulse ends in the rear baseline.
- `PSD_ERR_ENDS_LATE` : fraction of events where the pulse ends too late.
- `PSD_ERR_DUR_SHORT` : fraction of events with a pulse duration shorter than the specified minimum duration (specified in the algorithm parameter block). Typically, this parameter characterises the amount of noise triggers of a specific PSD channel. It should be monitored very carefully.
- `PSD_ERR_DUR_LONG` : fraction of events with a pulse duration longer than the specified maximum duration (specified in the algorithm parameter block).
- `POST_ERR_FRACT` : fraction of events with pulse shape analysis onground error. The following 9 columns give a breakdown of the onground analysis error types.
- `POST_ERR_PSD_ERR` : fraction of events with onboard analysis error.
- `POST_ERR_NO_PARA` : fraction of events for which no parameters are available for onground post-processing.
- `POST_ERR_NO_CONF` : fraction of events for which no PSD configuration information has been found for PSD science word decoding.
- `POST_ERR_NO_PRP` : fraction of events for which no PRP data has been available for onground post-processing.
- `POST_ERR_NO_COR` : fraction of events for which no COR data has been available for onground post-processing.
- `POST_ERR_NO_OSM` : fraction of events for which no OSM data has been available for onground post-processing.
- `POST_ERR_BAD_TTP1` : fraction of events with an invalid TTP1 value.
- `POST_ERR_BAD_TTP2` : fraction of events with an invalid TTP2 value.
- `POST_ERR_BAD_END` : fraction of events with an invalid energy value.
- `FIT_ERR_FRACT` : fraction of PSD curves with onground fit error.
- `FIT_ERR_PSD_ERR` : fraction of PSD curves with onboard analysis error.
- `FIT_ERR_NO_CURVE` : fraction of PSD curves for which no curve data was available for onground fitting.

- **FIT_ERR_FEW_STEPS** : fraction of PSD curves for which too few time steps were selected for onground fitting.
- **FIT_ERR_MANY_STEPS** : fraction of PSD curves for which too many time steps were selected for onground fitting.
- **FIT_ERR_MANY_TPLS** : fraction of PSD curves for which too many templates were selected for onground fitting.
- **FIT_ERR_START_LATE** : fraction of PSD curves which start too late for onground fitting to be performed correctly.
- **FIT_ERR_BASE_NEG** : fraction of PSD curves with negative baseline, preventing reliable onground fitting to be performed.
- **FIT_ERR_PA_NEG** : fraction of PSD curves with negative pulse area, preventing reliable onground fitting to be performed.
- **PE_NUM** : total number of PSD events.
- **CRVE_NUM** : total number of PSD curves.

Note that except of **OBT_START**, **OBT_STOP**, and **ONTIME**, all columns are vector columns of dimension 19.

Optionally, **spi_psd_performance** is able to generate ISDC alerts if some of the performance indicators fall out of defined limits. The alert limits are defined by an **SPI.-ALRT-LIM** structure from which the following columns are used by **spi_psd_performance** for alert generation:

- **PAR_NAME** specifies the parameter for which the limits apply (see the list above for valid parameter names). If one of the above parameter names is defined, the specified limits apply to **all** 19 PSD detection channels. By adding **_Ln** to the parameter name (where *n* runs from 0 to 18), parameter limits may be specified for a given PSD detection channel (for example **AVG_NOISE_L11** specifies the limits for the average noise for PSD channel 11). Channel specific parameters have precedence over common parameters (i.e. those without the **_Ln** extension), hence one may define common parameter limits for all 19 PSD channels and overwrite a few limits for specific channels by specifying explicitly a channel limit.
- **MIN_VAL** specifies the lower parameter limits (inclusive) for the four ISDC alert levels (DAL table columns 1-3 corresponds to alert levels 0-3).
- **MAX_VAL** specifies the upper parameter 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 performance parameters validity time interval. If the performance parameters validity time interval stops before the validity of the earliest alert limits, the earliest alert limits are used by **spi_psd_performance** (a warning is issued by **spi_psd_performance** 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_performance** (a warning is issued by **spi_psd_performance** in this case). If the performance parameters 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 performance parameters validity time interval.

5 Algorithm

Parameter averages are calculated using

$$\text{AVG_PARAMETER} = \frac{\sum_{i=1}^N \delta_{\text{NOERR}} \times \text{PARAMETER}_i}{\sum_{i=1}^N \delta_{\text{NOERR}}} \quad (1)$$

where $\delta_{\text{NOERR}} = 1$ if no PSD algorithm error occurred (error code equal to `PSDLIB_ERRCODE_NO_ERROR`; see `spi_psdlib` User Manual), and $\delta_{\text{NOERR}} = 0$ in case of an error (N is the number of parameter values `PARAMETERi`). In other words, only parameters are averaged for which no PSD algorithm error has occurred. For all parameters except of `AVG_CHISQR` the `PSD_ERR` field in the `SPI.-xCRV-PRP` HDU is used for error determination. Only for `AVG_CHISQR` the `PSD_FIT_ERR` field of the `SPI.-xCRV-PRP` HDU is considered. If the denominator of the above equation is zero, the average parameters is also set to zero.

Random mean scatters of parameters are calculated using

$$\text{RMS_PARAMETER} = \sqrt{\frac{\sum_{i=1}^N \delta_{\text{NOERR}} \times (\text{PARAMETER}_i^2 - \text{AVG_PARAMETER}^2)}{(\sum_{i=1}^N \delta_{\text{NOERR}}) - 1}} \quad (2)$$

If the denominator of the above equation is zero, the random mean scatter is set to zero. For all parameters except of `RMS_CHISQR` the `PSD_ERR` field in the `SPI.-xCRV-PRP` HDU is used for error determination. Only for `RMS_CHISQR` the `PSD_FIT_ERR` field of the `SPI.-xCRV-PRP` HDU is considered.

Fractions (parameters `PSD_ERR_FRACT`, `POST_ERR_FRACT`, and `FIT_ERR_FRACT` are calculated using

$$\text{FRACT} = \frac{\sum_{i=1}^N \delta_{\text{ERR}}}{\sum_{i=1}^N \delta_{\text{ERR}}} \quad (3)$$

where $\delta_{\text{ERR}} = 1$ if a specific PSD error class has occurred, and $\delta_{\text{ERR}} = 0$ otherwise. If the denominator of the above equation is zero, the fraction is set to zero. The three PSD error classed that are considered by `spi_psd_performance` are

- all onboard errors, i.e. `PSD_ERR != PSDLIB_ERRCODE_NO_ERROR`. The result is stored in the column `PSD_ERR_FRACT`.
- all onground errors, i.e. `PSD_CORFLAG != PSDLIB_POSTPROCESS_SINGLE && PSD_CORFLAG != PSDLIB_POSTPROCESS_MULTIPLE`. The result is stored in the column `POST_ERR_FRACT`. Only events are counted for which `COR` information is available.
- all onground fitting errors, i.e. `PSD_FIT_ERR != PSDLIB_ERRCODE_NO_ERROR`. The result is stored in the column `FIT_ERR_FRACT`.

Optional alert 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

$$\text{PARAMETER}_i < \text{MIN_VAL} \quad (4)$$

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

$$\text{PARAMETER}_i > \text{MAX_VAL} \quad (5)$$

is fulfilled.

Limit violation alerts for parameters based on PSD events (PE) (i.e. for the parameters `PSD_ERR_FRACT` and `POST_ERR_FRACT`) are only generated if the minimum number of PSD events for the respective channel is at least equal to the task parameter `minPE`, i.e.

$$\text{PE_NUM} \geq \text{minPE}. \quad (6)$$

Note that `PE_NUM` includes all PSD events, also those with PSD errors!

Limit violation alerts for all other parameters are only generated if

$$\text{CRVE_NUM} \geq \text{minCRVE} \quad (7)$$

is fulfilled, where again `CRVE_NUM` includes all PSD curves, also those with errors.

6 Alerts

`spi-psd_performance` 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 $n=0-18$ specifies the PSD detection channel 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
AVG_STARTTIME	0-3	standard
AVG_ENDTIME	0-3	standard
AVG_DURATION	0-3	standard
AVG_BASELINE	0-3	standard
AVG_BASE_SLOPE	0-3	standard
AVG_NOISE	0-3	standard
AVG_CHISQR	0-3	standard
RMS_STARTTIME	0-3	standard
RMS_ENDTIME	0-3	standard
RMS_DURATION	0-3	standard
RMS_BASELINE	0-3	standard
RMS_BASE_SLOPE	0-3	standard
RMS_NOISE	0-3	standard
RMS_CHISQR	0-3	standard
PSD_ERR_FRACT	0-3	standard
PSD_ERR_INVALID	0-3	Report to SPI PI (possible SPI/PSD configuration error).
PSD_ERR_NO_CORR	0-3	Report to SPI PI (possible PSD-DFEE-DPE data link problem).
PSD_ERR_NO_CONF	0-3	Possible missing previous SWG during <code>dp_spi-psd</code> execution
PSD_ERR_NO_INX2TTP	0-3	Check that <code>dp_spi-psd</code> has been run with correct parameters.
PSD_ERR_BAD_DETE	0-3	Report to SPI PI (possible PSD hardware problem).
PSD_ERR_BAD_DECODE	0-3	standard
PSD_ERR_SAT	0-3	standard
PSD_ERR_PA_SMALL	0-3	standard
PSD_ERR_PA_LARGE	0-3	standard
PSD_ERR_ATTP_EARLY	0-3	standard
PSD_ERR_ATTP_LATE	0-3	standard
PSD_ERR_BASE_LOW	0-3	standard
PSD_ERR_BASE_HIGH	0-3	standard
PSD_ERR_BASE_START	0-3	standard
PSD_ERR_BASE_END	0-3	standard
PSD_ERR_ENDS_LATE	0-3	standard
PSD_ERR_DUR_SHORT	0-2	standard
	3	Report to SPI PI (possible PSD hardware problem).
PSD_ERR_DUR_LONG	0-3	standard

Parameter	Level	Action
POST_ERR_FRACT	0-3	standard
POST_ERR_FRACT	0-3	standard
POST_ERR_PSD_ERR	0-3	standard
POST_ERR_NO_PARA	0-3	standard
POST_ERR_NO_CONF	0-3	standard
POST_ERR_NO_PRP	0-3	standard
POST_ERR_NO_COR	0-3	standard
POST_ERR_NO_OSM	0-3	standard
POST_ERR_BAD_TTP1	0-3	standard
POST_ERR_BAD_TTP2	0-3	standard
POST_ERR_BAD_END	0-3	standard
FIT_ERR_FRACT	0-3	standard
FIT_ERR_PSD_ERR	0-3	standard
FIT_ERR_NO_CURVE	0-3	standard
FIT_ERR_FEW_STEPS	0-3	standard
FIT_ERR_MANY_STEPS	0-3	standard
FIT_ERR_MANY_TPLS	0-3	standard
FIT_ERR_START_LATE	0-3	standard
FIT_ERR_BASE_NEG	0-3	standard
FIT_ERR_PA_NEG	0-3	standard

7 Error codes

The following error codes are defined:

SPI_PSD_PERFORMANCE_ERROR_MEM_ALLOC	-231800
SPI_PSD_PERFORMANCE_ERROR_INDEX_SELECT	-231801

They have the following meaning:

- **SPI_PSD_PERFORMANCE_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 numbers **SPI_PSD_PERFORMANCE_BUFFER_PE** and **SPI_PSD_PERFORMANCE_BUFFER_CRVE** in the **spi_psd_performance.h** header file and recompile the code.
- **SPI_PSD_EFFICIENCY_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 alert limit index you specified on input is somehow corrupted.

In addition, all errors that may occur in calls to ISDC support functions (such as for example DAL, RIL or PIL) are forwarded.