PACS User's Reference Manual

Herschel Data Processing

Version 3.0, Document Number: HERSCHEL-HSC-DOC-0935
06 May 2010

PACS User's Reference Manual: Herschel Data Processing

Table of Contents

Categorized view of Commands

This chapter provides a categorized view of all built-in DPfunctions, tasks and objects.

- **PACS**
  - **Calibration**
    - getCalPool - Get the default Calibration Pool.
    - getCalProduct - Get a specific version for a calibration product.
    - getCalStorage - Get the default calibration pool as a product storage.
  - **Calibration Framework**
    - getCalTree - Task to retrieve the PACS calibration tree.
    - **Common**
      - ChopperAngle - This calibration product defines the calibration for the chopper position readouts versus the chopper angle.
      - ChopperAngleRedundant - This calibration product defines the calibration for the chopper position readouts versus the chopper angle in the redundant mode.
      - ChopperJitterThreshold - This calibration product defines the thresholds for the required accuracy of the
    - **Spectrometer**
      - ArrayInstrument - This calibration product provides the array to instrument coordinate conversion.
  - **Spectrometer**
    - **Pipeline**
      - specProject - Projects the modules from the spectrometer onto a regular grid on the sky.
    - **Task**
      - SpecFitSignalDriftTask - Computes how the pixel response changes with time
- **Task**
  - **PACS Task**
    - AddInstantPointingTask - Include s/c pointing as coordinates and additonal pointing information like scanNumber in PACS frames product.
    - DecodeCapacitanceTask - This task decodes CRECR status and put the capacitance in pF either into
    - FilterOnTargetTask - unknown
    - FilterSlewTask - unknown
    - GetPointingProductTask - Retrieves the correct pointing product for a given sequence file or observation context
    - MakePacsPointingProductTask - unknown
    - SpecAddInstantPointingRampsTask - Add PointingProduct to Ramps class
- **class**

- Detection
- Ext - This class gives a representation of successive maxima as a chained structure
- HolderList
- PhotDiffCStoringTask
- PhotDriftCorrectionTask
- PhotRemoveMedianTask - This task is used with scan MAp AOR, where no chopping exists
- PhotTrendCSTask
- Reconstruction
- Wavelet
- WTMMLDeglitchingTask

pacs

spg

phot

- PhotConvDigit2VoltsTask - Convert Digits to Volts
- PhotCSProcessingTask - Jython task process calibration blocks found in the telemetry
- PhotFlagAdcSaturationTask - Jython task that sets the flags for saturated readouts
- PhotFlagBadPixelsTask - Jython task that sets the flags for bad pixels ("BAD_PIXEL" mask)
- PhotFlagClSaturationTask - Jython task that sets the flags for saturated readouts
- PhotFlagSaturationTask - Jython task that sets the flags for saturated readouts (populates SATURATION_LOW and SATURATION_HIGH mask).
- PhotProjectPointSourceTask - Project Photometer PointSource image

toolboxes

common

- ReadSimulatorDataTask - Read Simulator FITS files and import the data into Frames class

spg

- ConvXyStage2PointingTask - Add the XyStage "Pointing Information" ("Stage_X" and "Stage_Y" in mm) to Frames class
- PhotTreatCSTask - Treat Calibration Blocks for flux calibration and trend analysis

How to use this manual

The *User's Reference Manual* contains information about all the main tasks and classes that you can use within your scripts. There are four version of this manual: the *HCSS* version describes the functions provided by the core Herschel software, and is always shipped with HIPE; the three other versions describe functions provided by HIFI, PACS and SPIRE software. You may or may not have these additional manuals, depending on the software you installed.

This manual includes the following sections:

- Categorized view of Commands. All the functions described in the manual, organised by category.
- Section 1.1. This section lists all the available commands in alphabetical order. Each command has a description, usage instructions and examples.

II.1. Related documentation

The aim of this manual is to provide reference information for all the user-relevant aspects of the PACS software. Despite our efforts, you may find that some routines are missing, or have incomplete or inaccurate descriptions. In this case you are encouraged to consult the *Javadoc* developer's reference documentation. You can access the Javadoc by clicking on *PACS Developer's Reference Manual (API)* in the table of contents of the HIPE Help System.

Guidance on how to use the Javadoc is provided in the *Scripting and Data Mining* guide: ????.

Some Javadoc pages may have links to more in-depth developer documentation. Be aware that these are *not* fully fledged help documents and are most useful to system developers or advanced users only.

Inspecting the source code

If you are an advanced user versed in Java and Jython, you might want to have a look at the source code of a task or class. You can include source code in your HIPE installation in the following ways:

- If you are installing a developer build via the Continuous Installation System, the `--src=yes` option will install source code. This is enabled by default if you use the `--developer` option.

  With the `--unpack=yes` option, the source code will be unpacked into a `src` subdirectory in your HIPE installation. With the `--unpack=no` option, source files for each module will be kept as ZIP files in the repository (usually under `$HOME/.hcss.d/repository` in UNIX systems and under `%HOMEPATH%/.hcss.d/repository` in Windows).

- If you are using an installer, select the checkbox next to the question *Would you like to have the source code installed?* This is only available if you choose the *Advanced* installation. The source code will be in the `src` subdirectory of your HIPE installation.

Chapter 1. DP Commands

1.1. Introduction

This chapter is a complete listing of all built-in DP functions, task and objects, collectively referred to as commands.

How to use this chapter

All of Dp's commands are documented alphabetically in this chapter. A first section gives a general overview what a command is used for and how to use it. This includes a general description and some examples. The "API summary" section provides a quick overview of the available constructors, methods and properties of a command. It is intended as quick reference for users that just need to remember a certain functionality. The "API details" section provides more detailed information about the constructors, methods and properties. Furthermore it provides links to other references and information about the command's history.

Overview

A table gives the full name of the command, indicates if it is written in Java or Jython and if it is just an ordinary command or if it follows the HCSS Task specifications. The import statement in this table can be copied to a Jython console.

Description

General description of the command.

Examples

Most commands include one or more examples that demonstrates how the command is used. Most of the examples are just one or two lines of DP code that can be entered at the Jython prompt. Others are code fragments or routines designed to serve as an examples for your own programs.

Limitations and Miscellaneous

The "Limitations" and "Miscellaneous" sections describes the boundaries of applicability and other specialties of the command.

API Summary

The API Summary provides a quick summary of the commands constructor, method, and properties. Note that most of the arguments are positional parameters that must be supplied in the order indicated by the command's syntax. However arguments can be often supplied by specifying their names. If its the case then they can provided in any order.

Constructor

Constructors are used to create a command.

Method

Methods are functions of a command that can be executed.

Properties

Properties are fields of a command that can be read or written by a command. INPUT properties contain values that are required by a command, OUTPUT properties contain the result of a command, INOUT provide both functionalities at the same time.

See also

The "See also" section provides links to related commands, to other manuals, or to external resources in the Internet.

History

The "History" section describes the changes occurred to the command.

1.2. AbstractExtractionTask

Full Name:	herschel.pacs.spg.spec.AbstractExtractionTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import AbstractExtractionTask

Description

AbstractExtractionTask is an abstract Task to extract a spectrum or reconstruct a cube from a PACS cube.

AbstractExtractionTask is an abstract Task to extract a spectrum or reconstruct a cube from a PACS cube.

API Summary

Properties
PacsCube **cube** [INPUT, MANDATORY, default=No default value]
Boolean **oversample** [INPUT, OPTIONAL, default=Default value : true]
double **upsample** [INPUT, OPTIONAL, default=Default value : 3.0]
PacsCal **calTree** [INPUT, OPTIONAL, default=Default value : null]
OPTIONAL **specProperties** [SpecProperties, Default value : null, default=no default value]

API details

Properties

PacsCube cube [INPUT, MANDATORY, default=No default value]

The input PACS cube. *

Boolean oversample [INPUT, OPTIONAL, default=Default value : true]

Indicates whether to oversample.

double upsample [INPUT, OPTIONAL, default=Default value : 3.0]

The upsample factor.

PacsCal calTree [INPUT, OPTIONAL, default=Default value : null]

The PACS calibration tree.

OPTIONAL specProperties [SpecProperties, Default value : null, default=no default value]

The PACS spectrometer properties.

1.3. ActivateMasksTask

Full Name:	herschel.pacs.spg.ActivateMasksTask
Type:	Java Task -
Import:	from herschel.pacs.spg import ActivateMasksTask

Description

Task that set the status of a submitted masklist to active.

API Summary

Jython Syntax
`out = activateMasks(in ,["BLINDPIXELS", "SATURATION"])`

Properties
PhotRaw **in : PacsProduct (Frames** [Ramps, PacsCube) that contains masks, default=no default value]
type **masks: List of masknames. The specified masks will be set to active** [INPUT, MANDATORY, default=no default value]
type **activate: true: the specified masks will be set to active (default)** [false: the masks will be deactivated. Alternatively you may use also the task DeactivateMasksTask, MANDATORY, default=no default value]
type **exclusive: if True** [the specified Masks will be set as activate commands. All others will be set to the complementary state., MANDATORY, default=no default value]
type **out : the same PacsProduct as in but with changed mask status** [INPUT, MANDATORY, default=no default value]

API details

Properties

PhotRaw in : PacsProduct (Frames [Ramps, PacsCube) that contains masks, default=no default value]

type masks: List of masknames. The specified masks will be set to active [INPUT, MANDATORY, default=no default value]

type activate: true: the specified masks will be set to active (default) [false: the masks will be deactivated. Alternatively you may use also the task DeactivateMasksTask, MANDATORY, default=no default value]

type exclusive: if True [the specified Masks will be set as activate commands. All others will be set to the complementary state., MANDATORY, default=no default value]

if False, the state of other masks is not touched. Default value is False.

```
type out : the same PacsProduct as in but with changed mask
status [INPUT, MANDATORY, default=no default value]
```

1.4. AddInstantPointingTask

Full Name:	herschel.pacs.spg.AddInstantPointingTask
Type:	Java Task -
Import:	from herschel.pacs.spg import AddInstantPointingTask
Category:	Task/PACS Task

Description

Include s/c pointing as coordinates and additonal pointing information like scanNumber in PACS frames product.

By default the Filtered Pointing information is used, but also the gyro propagated Pointing information may be used.

This is done by using the Frames status entry FINETIME and extract the associated information from the PointingProduct.

Also the SIAM matrix is applied and aberration is done (if the proper Products are passed)

The result is added to the status entry of the Frames Product

API Summary

Jython Syntax
`outFrames = photAddInstantPointing(inFrames, pp [calTree] [siam] [orbitEphem] [horizons] [noInter] [useGyro] [copy])`

Properties
Frames **inFrames** [INPUT, MANDATORY, default=NO default value]
PointingProduct **pp** [INPUT, MANDATORY, default=NO default value]
PacsCal **calTree** [INPUT, OPTIONAL, default=None]
Siam **siam** [INPUT, OPTIONAL, default=None]
OrbitEphemerisProduct **orbitEpehm** [INPUT, OPTIONAL, default=None]
Horizons **horizons** [INPUT, OPTIONAL, default=None]
Boolean **useGyro** [INPUT, OPTIONAL, default=False]
Integer **copy** [INPUT, OPTIONAL, default=None]
Frames **outFrames** [OUTPUT, MANDATORY, default=NO default value Output Frames]

API details

Properties

Frames inFrames [INPUT, MANDATORY, default=NO default value]
Input Frames class

PointingProduct pp [INPUT, MANDATORY, default=NO default value]
Pointing Product

PacsCal calTree [INPUT, OPTIONAL, default=None]

Pacs Calibration Tree

Siam siam [INPUT, OPTIONAL, default=None]

SIAM Product

OrbitEphemerisProduct orbitEpehm [INPUT, OPTIONAL, default=None]

Orbit Ephemeris Product needed for aberration correction of non SSO objects

Horizons horizons [INPUT, OPTIONAL, default=None]

Horizons Product (Needed for aberration correction of SSO objects)

Boolean useGyro [INPUT, OPTIONAL, default=False]

Use the Gyro propagated information instead of the filtered

Integer copy [INPUT, OPTIONAL, default=None]

Copy the Frames class instead of just adding the status word

Frames outFrames [OUTPUT, MANDATORY, default=NO default value Output Frames]

Output frames (default it is the reference of input Frames + new status entries)

1.5. AddObcp2Frames

Full Name:	herschel.pacs.spg.level0.AddObcp2Frames
Type:	Java Class -
Import:	from herschel.pacs.spg.level0 import AddObcp2Frames

Description

Add the OBCP numbers to the Meta data of a Frames class

API Summary

Jython Syntax
`Frames outFrames = addOBCP2Frames(Frames inFrames, PacketSequence seq [, int copy = <copy>])`

Properties
`Frames` **`inFrames`** `[INPUT, MANDATORY, default=NO default value]`
`Frames` **`outFrames`** `[OUTPUT, MANDATORY, default=NO default value]`
`String` **`seq`** `[INPUT, MANDATORY, default=No default value]`
`Integer` **`copy`** `[INPUT, Optional, default=default value: 0]`

API details

Properties

`Frames inFrames [INPUT, MANDATORY, default=NO default value]`

Frames input

`Frames outFrames [OUTPUT, MANDATORY, default=NO default value]`

Output Frames with OBCP data

`String seq [INPUT, MANDATORY, default=No default value]`

PacketSequence containing the SPEC_HK /PHOT_HK belonging to the period of the Frames class

`Integer copy [INPUT, Optional, default=default value: 0]`

indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

History

- 1.5 2008/09/10 jdejong Generalized AddObcp2Frames to work for ramps as well
- 1.4 2008/05/26 jdejong Improved exception handling
- 1.3 2007/09/19 ewieprec just a test
- 1.2 2007/04/30 ewieprec java version for E2E

- 1.1 2007/04/30 ewieprec java version for E2E
- 1.0 30-Apr-2007 EkW Converted from JYTHON version

1.6. AddObcp2FramesTask

Full Name:	herschel.pacs.spg.level0.AddObcp2FramesTask
Type:	Java Task -
Import:	from herschel.pacs.spg.level0 import AddObcp2FramesTask

Description

addObcp2Frames:

Add the OBCP numbers to the Meta data of a Frames class

API Summary

Jython Syntax
`Frames outFrames = addOBCP2Frames(Frames inFrames, PacketSequence seq [, int copy = <copy>])`

Properties
`Frames inFrames [INPUT, MANDATORY, default=NO default value]`
`Frames outFrames [OUTPUT, MANDATORY, default=NO default value]`
`String seq [INPUT, MANDATORY, default=No default value]`
`Integer copy [INPUT, Optional, default=default value: 0]`

API details

Properties

`Frames inFrames [INPUT, MANDATORY, default=NO default value]`

Frames input

`Frames outFrames [OUTPUT, MANDATORY, default=NO default value]`

Output Frames with OBCP data

`String seq [INPUT, MANDATORY, default=No default value]`

PacketSequence containing the SPEC_HK /PHOT_HK belonging to the period of the Frames class

`Integer copy [INPUT, Optional, default=default value: 0]`

indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

History

- `2007-04-30 - EkW`: 1.0 Converted from JYTHON version
- `2008-09-10 - JdJ`: 1.3 Generalized this task for all PACS products

1.7. AddObcp2FramesTask

Full Name:	herschel.pacs.toolboxes.spg.AddObcp2FramesTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spg import AddObcp2FramesTask

Description

Add the OBCP numbers to the Meta data of a Frames class

API Summary

Jython Syntax
`Frames outFrames = addOBCP2Frames(Frames inFrames, PacketSequence seq [, int copy = <copy>])`

Properties
`Frames inFrames [INPUT, MANDATORY, default=NO default value]`
`Frames outFrames [OUTPUT, MANDATORY, default=NO default value]`
`String seq [INPUT, MANDATORY, default=No default value]`
`Integer copy [INPUT, Optional, default=default value: 0]`

API details

Properties

`Frames inFrames [INPUT, MANDATORY, default=NO default value]`

Frames input

`Frames outFrames [OUTPUT, MANDATORY, default=NO default value]`

Output Frames with OBCP data

`String seq [INPUT, MANDATORY, default=No default value]`

PacketSequence containing the SPEC_HK /PHOT_HK belonging to the period of the Frames class

`Integer copy [INPUT, Optional, default=default value: 0]`

indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

History

- 1.0 13-Apr-2006 EkW Initial version of this Task
- 1.1 20-Jul-2006 Ekw Work around for bug in BinaryStructures

1.8. AddUtcTask

Full Name:	herschel.pacs.spg.AddUtcTask
Type:	Java Task -
Import:	from herschel.pacs.spg import AddUtcTask

Description

Correction of Time difference between the on board time and ground UTC using the time correlation file. A new status column Utc is added.

API Summary

Jython Syntax
`outFrames = addUtc(inFrames, timeCorr)`

Properties
`Frames inFrames [INPUT, MANDATORY, default=NO default]`
`TimeCorrProduct timeCorr [INPUT, MANDATORY, default=NO default]`
`Frames outFrames [OUTPUT, MANDATORY, default=NO default value Output Frames]`

API details

Properties

`Frames inFrames [INPUT, MANDATORY, default=NO default]`
Frames Producrt Input Frames class

`TimeCorrProduct timeCorr [INPUT, MANDATORY, default=NO default]`
Time Correlation Product holding the time corrections for UTC against OBT

`Frames outFrames [OUTPUT, MANDATORY, default=NO default value Output Frames]`
Output frames (default it is the reference of input Frames + new status entries)

1.9. ArrayInstrument

Full Name:	herschel.pacs.cal.spectrometer.ArrayInstrument
Type:	Java Class -
Import:	from herschel.pacs.cal.all import *
Category:	PACS/Calibration Framework/Spectrometer

Description

This calibration product provides the array to instrument coordinate conversion.

The content can be accessed using the *blue* and *red* methods.

1.10. AverageFramesTask

Full Name:	herschel.pacs.toolboxes.common.AverageFramesTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import AverageFramesTask

Description

This Jython task is averaging Frame data in the time dimension over nAvg values.

- Frame Signals (double) are just averaged
- Frame Wavelengths (double) are just averaged
- The Frame Masks (boolean) are reduced. Optional user may reduce this data by "AND" or "OR" operations. - not yet implemented
- The Frame Status (int) are averaged. But the then rounded to the nearest Integer.
- The Frame Status (boolean) are treated as Frame Masks
- The Frame Status (string) are selected by majority, in case of no majority the first one is taken

API Summary

Jython Syntax
`Frames oframes = averageFrames(Frames iframes, int avgNr, String boolOP , [,copy=0] )`

Properties
`Frames iframes [INPUT, MANDATORY, default=NO default value]`
`String avgNr [INPUT, MANDATORY, default=NO default value]`
`String boolOp [INPUT, MANDATORY, default=NO default value]`
`String copy [INPUT, OPTIONAL, default=default value : 0]`
`Frames oframes [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

Frames iframes [INPUT, MANDATORY, default=NO default value]
input Frames object

String avgNr [INPUT, MANDATORY, default=NO default value]
Number of readouts to average. Is the input not a multiple of avgNr, the additional samples are ignored

String boolOp [INPUT, MANDATORY, default=NO default value]
How to deal with Boolean values in Mask and Status. This function is not yet active. Currently we return True if one or more True is set within average interval

String copy [INPUT, OPTIONAL, default=default value : 0]
Copy the input (0) and generate new output or overwrite the input (1)

Frames oframes [OUTPUT, MANDATORY, default=NO default value]
output Frames object

History

- 0.1 20060317 EkW initial version derived from JAVA version
- TODO : activate boolOps
- Currently we return True if one or more True is set within average interval

1.11. AverageRampsTask

Full Name:	herschel.pacs.toolboxes.spg.AverageRampsTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spg import AverageRampsTask

Description

Jython function that averages a Ramps object

Simulates what the on-board reduction software does: average the readouts per nSamplesToAverage

API Summary

Jython Syntax
`Ramps outRamps = averageRamps(Ramps inRawRamps, int nSamplesToAverage)`

Properties
`Ramps inRawRamps [INPUT, MANDATORY, default=NO default value]`
`int nSamplesToAverage [INPUT, MANDATORY, default=NO default value]`
`Ramps outRamps [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

Ramps inRawRamps [INPUT, MANDATORY, default=NO default value]
input Ramps to be averaged

int nSamplesToAverage [INPUT, MANDATORY, default=NO default value]
number of samples to average

Ramps outRamps [OUTPUT, MANDATORY, default=NO default value]
averaged Ramps with one value per nSamplesToAverage

History

- 1.0 20060707 JV initial version
- 1.1 20060830 JV added status and updated averaged signal
- 1.2 20060905 JV changed parameter types to IN/OUT

1.12. Band

Full Name:	herschel.pacs.signal.Band
Type:	Java Class -
Import:	from herschel.pacs.signal import Band

History

- 13 jan 2009 : first version

1.13. calcGaps

Full Name:	herschel.pacs.toolboxes.common.calcGaps
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import calcGaps

Description

Calculate missing packets from te SourceSequenceCounter per APID and event packages :

['DPU_HK_LOST','DPU_EVENT_LOST','DPU_GEN_TM_LOST','DPU_TC_LOST']

API Summary

Jython Syntax
`table = calcGaps(seq, interval)`

Properties
`PacketSequence seq [INPUT, MANDATORY, default=NO default value PacketSequence]`
`int interval [INPUT, MANDATORY, default=NO default missing packets out of in seconds]`
`TableDataset table [OUTPUT, MANDATORY, default=NO default value table containing the result]`

API details

Properties

`PacketSequence seq [INPUT, MANDATORY, default=NO default value PacketSequence]`

`int interval [INPUT, MANDATORY, default=NO default missing packets out of in seconds]`

`TableDataset table [OUTPUT, MANDATORY, default=NO default value table containing the result]`

Table columns : Time StartTime EndTime [microsec since 1988] Gaps [Number of missing packets] Meta data : 'DPU_HK_LOST','DPU_EVENT_LOST','DPU_GEN_TM_LOST','DPU_TC_LOST' To print the time you need to : from herschel.share.fltdyn.time imort * print FineTime(t["Time"].data[0])

History

- 20040921 EkW version 1.0 first version without the DataRate around the telemetry drops
- 20040928 EkW version 1.1 correct documentation, add type to table RELATION
- 20040929 EkW version 1.2 bug fix, now working per APID , rather then per packet type

- 20051103 EkW version 2.0 copied from TMGaps.py
- 20051107 JS version 2.1 clean-up of imports

1.14. CAP_0.7.6

Full Name:	herschel.pacs.toolboxes.cap.CAP_0.7.6
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.cap import CAP_0.7.6

Description

draft script for test analysis procedure 0.7.6

synchronous operation of chopper with detectors (incl. reproducibility aspects)-Spectroscopy

See also

- ???

History

- 20040526 JS 1.1
- 20040526 JS 1.2 debugged chopper plateau search, refurbished due to new IA numerics package
- 20040528 JS 1.3 new handling of indices (Selection objects)
- 20040601 JS 1.4 debugged according to new IA numerics
- 20040702 JS 1.5 improved handling if no set_time found
- 20040716 JS 1.6 step no. 2.04 (Synchronization check II) added
- 20040727 JS 1.7 improved signal indices handling
- 20040727 JS 1.8 typo corrected
- 20040730 JS 1.9 seq.getMeasures() -> seq.getRawMeasures()
- 20040730 JS 1.10 select diagnostic HK packets
- 20041026 JS 1.11 adjusted according to CQM-tests on 13/09/04 at MPE (chopper performance tests)
- using files QILT_0.7.6_spec_00007_Chopper_performance_03-20.tm
- (only diagnostic HK data!)
- no detector data available!
- no LABEL information available!
- 20041028 JS 1.12 improved version
- 20050418 JS 1.13 adjusted according to toolbox concept
- 20050704 JS 1.14 adopted to new IA version, readtm() introduced, chopper voltage removed
- 20050728 JS 1.15 header converted to jtags framework
- 20050923 JS 1.16 All diagnostic HK keywords have changed from DMC_... to DM_...

- 20050928 JS 1.17 improved version
- 20060503 JS 1.18 chopper accuracy threshold dependency on deflection introduced
- 20060511 JS 1.19 convertChopperAngleSteps() used to get chopper accuracy threshold
- 20060620 JS 1.20 input for chopPos2FpuAnglePol converted to Double1d
- 20060628 JS 1.21 "FM" improved version, synchronous operation with detectors added
- 20080905 JS 1.3 Version updated for IST

1.15. CAP_1.2.11

Full Name:	herschel.pacs.toolboxes.cap.CAP_1.2.11
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.cap import CAP_1.2.11

Description

draft script for test analysis procedure 1.2.11

Linearity of CRE readout

See also

- ???

History

- 20040623 JS 1.1
- 20040702 JS 1.2 improved handling if no set_time found
- 20040705 JS 1.3 improved array handling
- 20040707 JS 1.4 Python list appending exchanged by Int1d appending
- 20040708 JS 1.5 determination of non-linearity according to St. Birkmanns diploma thesis added
- 20041210 JS 1.6 rewritten according to CRE linearity tests carried out
- at 20041011. The results are written in two files for
- the red and blue detectors, respectively.
- 20041216 JS 1.7 bug fix: factor of 256 was missing in the current noise density calculation
- 20050207 JS 1.8 improved version
- 20050418 JS 1.9 adjusted according to toolbox concept
- 20050629 JS 1.10 adopted to the Ramps concept
- 20050705 JS 1.11 automatic loop through available Ramps
- 20050728 JS 1.12 header converted to jtags framework
- 20060306 JS 1.13 getRampValues() -> getSignal()

1.16. CAP_1.2.16

Full Name:	herschel.pacs.toolboxes.cap.CAP_1.2.16
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.cap import CAP_1.2.16

Description

draft script for test analysis procedure 1.2.16

This script should establish the time constant: switch-on spectrometer

History

- 20040902 JS 1.1
- 20060306 JS 1.2
- 20060629 JS 1.3 upgrade according to new FM PTD

1.17. CAP_1.2.1_ist

Full Name:	herschel.pacs.toolboxes.cap.CAP_1.2.1_ist
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.cap import CAP_1.2.1_ist

Description

This script should establish the NEP for each Tdet and Vbias and each pixel to be able to determine the optimum bias and temperature settings of the detectors

See also

- req. 1.2.1, 1.2.2, 1.2.7, 1.2.10 and 1.2.17

History

- 20040405 JS 1.1
- 20040527 JS 1.2 refurbished according to new IA numerics package
- 20040528 JS 1.3 new handling of indices (Selection objects)
- 20040601 JS 1.4 debugged according to new IA numerics
- 20040604 JS 1.5 math.sqrt replaced by SQRT
- 20040702 JS 1.6 improved handling if no set_time found
- 20040705 JS 1.7 signal reading changed according to new OBSW using averaged subramps
- 20040726 JS 1.8 refurbished and significantly improved
- 20040730 JS 1.9 seq.getMeasures() -> seq.getRawMeasures()
- 20040730 JS 1.10 select diagnostic HK packets
- 20060713 JS 1.11 updated for FM ILT tests
- 20060714 JS 1.12 req. 1.2.2 added
- 20060725 JS 1.13 correct capacitance units
- 20060727 JS 1.14 minor change
- 20061005 JS 1.15 reworked to work on signal modulation
- 20090109 JS 1.16 adopted to dp-pacs environment

1.18. CAP_1.2.1

Full Name:	herschel.pacs.toolboxes.cap.CAP_1.2.1
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.cap import CAP_1.2.1

Description

This script should establish the NEP for each Tdet and Vbias and each pixel to be able to determine the optimum bias and temperature settings of the detectors

See also

- req. 1.2.1, 1.2.2, 1.2.7, 1.2.10 and 1.2.17

History

- 20040405 JS 1.1
- 20040527 JS 1.2 refurbished according to new IA numerics package
- 20040528 JS 1.3 new handling of indices (Selection objects)
- 20040601 JS 1.4 debugged according to new IA numerics
- 20040604 JS 1.5 math.sqrt replaced by SQRT
- 20040702 JS 1.6 improved handling if no set_time found
- 20040705 JS 1.7 signal reading changed according to new OBSW using averaged subramps
- 20040726 JS 1.8 refurbished and significantly improved
- 20040730 JS 1.9 seq.getMeasures() -> seq.getRawMeasures()
- 20040730 JS 1.10 select diagnostic HK packets
- 20060713 JS 1.11 updated for FM ILT tests
- 20060714 JS 1.12 req. 1.2.2 added
- 20060725 JS 1.13 correct capacitance units
- 20060727 JS 1.14 minor change
- 20061005 JS 1.15 reworked to work on signal modulation

1.19. CAP_1.2.3

Full Name:	herschel.pacs.toolboxes.cap.CAP_1.2.3
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.cap import CAP_1.2.3

Description

draft script for test analysis procedure 1.2.3

Dynamic range per integration capacitor and reset interval

See also

- ???

History

- 20050721 JS 1.1 initial version
- 20050728 JS 1.2 header converted to jtags framework
- 20060306 JS 1.3 getRampValues() -> getSignal()

1.20. CAP_1.2.4

Full Name:	herschel.pacs.toolboxes.cap.CAP_1.2.4
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.cap import CAP_1.2.4

Description

draft script for test analysis procedure 1.2.4

This script should investigate the CRE dynamic range/saturation level and linearity. Additionally, it should determine the optimal checkout voltage.

See also

- ???

History

- 20040708 JS 1.1
- 20041110 JS 1.2 rewritten according to checkout voltage tests carried out
- at 20040916. The results are written in two files for
- the red and blue detectors, respectively. The .tm files
- are read in using a loop. Adjust filename and the num
- array accordingly.
- 20041117 JS 1.3 debugged and improved version
- 20041119 JS 1.4 improved version
- 20050418 JS 1.5 adjusted according to toolbox concept
- 20050706 JS 1.6 adopted to the Ramps concept
- 20050728 JS 1.7 header converted to jtags framework
- 20060306 JS 1.8 getRampValues() -> getSignal()
- 20060718 JS 1.9 adopted for FM tests

1.21. CAP_2.3.1_aut

Full Name:	herschel.pacs.toolboxes.cap.CAP_2.3.1_aut
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.cap import CAP_2.3.1_aut

Description

draft script for test analysis procedure 2.3.1:

"Angular calibration of PACS chopper" chopper open loop operations: determine the damping time constant to get the specific torque, spring constant

See also

- 2.3.1

History

- 20060503 JS 1.1
- 20060612 JS 1.2 determination of specific torque implemented
- 20060620 JS 1.3 input for chopPos2FpuAnglePol converted to Double1d
- 20061110 JS 1.4 improved during FM ILT
- 20061207 JS 1.5 further improvements
- 20070109 JS 1.6 further improvements
- 20070123 JS 1.7 automatized version

1.22. CAP_2.3.1

Full Name:	herschel.pacs.toolboxes.cap.CAP_2.3.1
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.cap import CAP_2.3.1

Description

draft script for test analysis procedure 2.3.1:

"Angular calibration of PACS chopper" chopper open loop operations: determine the damping time constant to get the specific torque, spring constant

See also

- 2.3.1

History

- 20060503 JS 1.1
- 20060612 JS 1.2 determination of specific torque implemented
- 20060620 JS 1.3 input for chopPos2FpuAnglePol converted to Double1d
- 20061110 JS 1.4 improved during FM ILT
- 20061207 JS 1.5 further improvements
- 20070109 JS 1.6 further improvements

1.23. CAP_2.3.2

Full Name:	herschel.pacs.toolboxes.cap.CAP_2.3.2
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.cap import CAP_2.3.2

Description

draft script for test analysis procedure 2.3.2:

Duty Cycle of Waveforms

See also

- 2.3.2

History

- 20060421 JS 1.1
- 20060511 JS 1.2 convertChopperAngleSteps() used to get chopper accuracy threshold
- 20060511 JS 1.3 getDutyCycle call adjusted to additional input parameter
- 20060512 JS 1.4 bug fix
- 20060608 JS 1.5 improved version, considers also the zero offset
- 20060620 JS 1.6 input for chopPos2FpuAnglePol converted to Double1d
- 20061110 JS 1.7 improved during FM ILT

1.24. CAP_3.1.2

Full Name:	herschel.pacs.toolboxes.cap.CAP_3.1.2
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.cap import CAP_3.1.2

Description

This script should establish for a chop direction a chop throw on sky for chopping

between two commanded chopper readouts

See also

- req. 3.1.2

History

- 20060727 JS 1.1 Prototype

1.25. CAP_CPSpecGeGa_Utils_gratPos2Alpha

Full Name:	herschel.pacs.toolboxes.cap.CAP_CPSpecGeGa_Utils_gratPos2Alpha
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.cap import CAP_CPSpecGeGa_Utils_gratPos2Alpha

Description

Calculate the wavelength corresponding to a grating position

Evaluates the Littrow equation as described in : "Wavelength Calibration of the PACS Spectrometer AVM/CQM" ILT PTD 4.2.1 FGB, Issue 1.0, feb 4,2005, with a polynomial relation between grating position and alpha per pixel

API Summary

Jython Syntax

```
Double wave = gratPos2Wave(Double gratPos, String band, int pix,
int mod,
[, PacsCal calTree][, LittrowPolynomes littrowPolynomes]
[, model = "FM"][, time=Date()])

Double1d wave = gratPos2Wave(Double1d gratPos, String band, int
pix, int mod,
[, PacsCal calTree][, LittrowPolynomes littrowPolynomes]
[, model = "FM"][, time=Date()])
```

Properties
Double1d **gratPos** [INPUT, MANDATORY, default=NO default value]
String **band** [INPUT, MANDATORY, default=NO default value]
int **pix** [INPUT, MANDATORY, default=NO default value]
int **mod** [INPUT, MANDATORY, default=NO default value]
PacsCal **calTree** [INPUT, OPTIONAL, default=None]
LittrowPolynomes **littrowPolynomes** [INPUT, OPTIONAL, default=None]
String **model** [INPUT, Optional, default=None]
FineTime **time** [INPUT, Optional, default=None]
Double1d **wave** [OUTPUT, MANDATORY, default=NO default value]

API details

Properties

Double1d gratPos [INPUT, MANDATORY, default=NO default value]

Grating position readback

String band [INPUT, MANDATORY, default=NO default value]

Spectral band: B3A : Blue short wave channel : filter A, order=3, 50-70um B2A : Blue short wave channel : filter A, order=2, 50-70um B2B : Blue long wave channel : filter B, order=2, 70-100um R1 : Red channel : order=1, 100-200um

int pix [INPUT, MANDATORY, default=NO default value]
Pixel number within a module (1,2,3 .., 16)

int mod [INPUT, MANDATORY, default=NO default value]
Module number (0,1,2,3 .., 24)

PacsCal calTree [INPUT, OPTIONAL, default=None]
cal file access

LittrowPolynomes littrowPolynomes [INPUT, OPTIONAL, default=None]
Specific version of LittrowPolynomes calibration table to use

String model [INPUT, Optional, default=None]
model name FM, FS or QM

FineTime time [INPUT, Optional, default=None]
time of measurement

Double1d wave [OUTPUT, MANDATORY, default=NO default value]
computed wavelength in um

See also

- ???
- ???

History

- 1.0 07-Mar-2005 EkW - Initial version as waveCalc
- Basing on BVs py script and FGBs Littrow equation
- 1.1 24-Mar-2005 EkW - Converted to Toolbox function
- 1.2 27-Apr-2005 BV - Changed interface and read parameters from calfile
- 1.3 04-Aug-2005 JS - header adopted to jtags syntax
- 1.4 18-Mar-2007 BV - Changed default version to FM_1_0
- 2.0 20-Nov-2007 EkW - adopt to new calibration framework
- 2.1 22-Feb 2008 JS - adapt again to new cal framework
- 3.0 18-Mar-2008 BV - Changed to use LittrowPolynomes, merged with 2.1
- 3.1 23-Apr-2009 BV - SPR PACS 1543 - don't read calfile if it is passed as
- an argument already

1.26. Channel

Full Name:	herschel.pacs.signal.Channel
Type:	Java Class -
Import:	from herschel.pacs.signal import Channel

History

- 13 Jan 2009 : first version

1.27. chopNodStarL1

Full Name:	herschel.pacs.spg.pipeline.ipipe.spec.chopNodStarL1
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.ipipe.spec import chopNodStarL1

History

- 1.0 20090313 JS initial version
- 1.1 20090624 JS updated
- 1.2 20090716 JS introduce slicing

1.28. chopNodStarL2

Full Name:	herschel.pacs.spg.pipeline.ipipe.spec.chopNodStarL2
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.ipipe.spec import chopNodStarL2

History

- 1.0 20090313 JS initial version
- 1.1 20090624 JS updated
- 1.2 20090716 JS introduce slicing

1.29. ChopperAngle

Full Name:	herschel.pacs.cal.common.ChopperAngle
Type:	Java Class -
Import:	from herschel.pacs.cal.all import *
Category:	PACS/Calibration Framework/Common

Description

This calibration product defines the calibration for the chopper position readouts versus the chopper angle.

The product and its content can be access starting from the calibration tree using the dot-notation, e.g. when `fm` is a calTree

```
za = fm.chopperAngle.zeissAmplification
```

1.30. ChopperAngleRedundant

Full Name:	herschel.pacs.cal.common.ChopperAngleRedundant
Type:	Java Class -
Import:	from herschel.pacs.cal.all import *
Category:	PACS/Calibration Framework/Common

Description

This calibration product defines the calibration for the chopper position readouts versus the chopper angle in the *redundant* mode.

The product and its content can be access starting from the calibration tree using the dot-notation, e.g. when `fm` is a calTree

```
za = fm.chopperAngleRedundant.zeissAmplification
```

1.31. ChopperJitterThreshold

Full Name:	herschel.pacs.cal.common.ChopperJitterThreshold
Type:	Java Class -
Import:	from herschel.pacs.cal.all import *
Category:	PACS/Calibration Framework/Common

Description

This calibration product defines the thresholds for the required accuracy of the

final chopper positions for the science and calibration window. The available methods are:

1.32. ChopPos2FpuAnglePolTask

Full Name:	herschel.pacs.spg.ChopPos2FpuAnglePolTask
Type:	Java Task -
Import:	from herschel.pacs.spg import ChopPos2FpuAnglePolTask

Description

Convert chopper digital units to chopper angle wrt FPU.

Convert the digital readback units of the chopper position to an angle (degrees).

API Summary

Jython Syntax
`fpuAngle = chopPos2FpuAnglePol(chopperPosition [,redundant] [,calTree] [,chopperAngle] [,chopperAngleRedundant][,model] [,time])`

Properties
`Int1d` **`chopperPosition`** `[INPUT, MANDATORY, default=NO default value]`
`Integer` **`redundant`** `[INPUT, OPTIONAL, default=default value: 0]`
`PacsCal` **`calTree`** `[INPUT, OPTIONAL, default=default value: null]`
`ChopperAngle` **`chopperAngle`** `[INPUT, Optional, default=default value: null]`
`ChopperAngleRedundant` **`chopperAngleRedundant`** `[INPUT, Optional, default=default value: null]`
`String` **`model`** `[INPUT, Optional, default=default value: "FM"]`
`FineTime` **`time`** `[INPUT, Optional, default=default value: present date]`
`Double1d` **`fpuAngle`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Int1d chopperPosition [INPUT, MANDATORY, default=NO default value]`

Chopper readback position

`Integer redundant [INPUT, OPTIONAL, default=default value: 0]`

if 0: nominal field plate calibration is used (FPI), if 1: redundant one is used (FPII)

`PacsCal calTree [INPUT, OPTIONAL, default=default value: null]`

cal file access

`ChopperAngle chopperAngle [INPUT, Optional, default=default value: null]`

chopper calibration readouts to fpu angle cal file access

ChopperAngleRedundant chopperAngleRedundant [INPUT, Optional, default=default value: null]

chopper calibration for redundant FP readouts to fpu angle cal file access

String model [INPUT, Optional, default=default value: "FM"]

model name FM, FS or QM

FineTime time [INPUT, Optional, default=default value: present date]

time of cal data

Double1d fpuAngle [OUTPUT, MANDATORY, default=NO default value]

Chopper physical angle (i.e. wrt FPU, not on the sky) in degrees

1.33. chopPos2SkyAngle

Full Name:	herschel.pacs.spg.chopPos2SkyAngle
Type:	Jython Task -
Import:	from herschel.pacs.spg import chopPos2SkyAngle

Description

Convert chopper digital units to chopper angle wrt to zero point on sky in arcmin

API Summary

Jython Syntax
`Double1d skyAngle = chopPos2SkyAngle(Int1d chopperPosition [, redundant = 0][CalibrationTree calTree][,ChopperSkyAngle chopperSkyAngle][,ChopperAngle chopperAngle] [,ChopperAngleRedundant chopperAngleRedundant] [,String model][,FineTime time])`

Properties
`Int1d` **`chopperPosition`** `[INPUT, MANDATORY, default=NO default value]`
`int` **`redundant`** `[INPUT, Optional, default=default value: 0]`
`CalibrationTree` **`calTree`** `[INPUT, Optional, default=default value: None]`
`ChopperSkyAngle` **`chopperSkyAngle`** `[INPUT, Optional, default=default value: None]`
`ChopperAngle` **`chopperAngle`** `[INPUT, Optional, default=default value: None]`
`ChopperAngleRedundant` **`chopperAngleRedundant`** `[INPUT, Optional, default=default value: None]`
`String` **`model`** `[INPUT, Optional, default=default value: "FM"]`
`FineTime` **`time`** `[INPUT, Optional, default=default value: present date]`
`Double1d` **`skyAngle`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Int1d chopperPosition [INPUT, MANDATORY, default=NO default value]`
Chopper readback position

`int redundant [INPUT, Optional, default=default value: 0]`
if 0: nominal field plate calibration is used (FPI), if 1: redundant one is used (FPII)

`CalibrationTree calTree [INPUT, Optional, default=default value: None]`
cal file access

ChopperSkyAngle chopperSkyAngle [INPUT, Optional, default=default value: None]
chopper fpu angle to sky angle cal file access

ChopperAngle chopperAngle [INPUT, Optional, default=default value: None]
chopper calibration readouts to fpu angle cal file access

ChopperAngleRedundant chopperAngleRedundant [INPUT, Optional, default=default value: None]
chopper calibration for redundant FP readouts to fpu angle cal file access

String model [INPUT, Optional, default=default value: "FM"]
model name FM, FS or QM

FineTime time [INPUT, Optional, default=default value: present date]
time of measurement

Double1d skyAngle [OUTPUT, MANDATORY, default=NO default value]
Chopper physical angle (i.e. wrt to optical zero on the sky) in arcmin

History

- 13/03/2007 JS Creation
- 22/11/2007 JS adaption to new cal framework
- 21/02/2008 JS adapt again to new cal framework
- 12/11/2008 JS remove zeroOffset as optional parameter, is in ChopperSkyAngle cal file now

1.34. CleanPlateauFramesTask

Full Name:	herschel.pacs.spg.CleanPlateauFramesTask
Type:	Java Task -
Import:	from herschel.pacs.spg import CleanPlateauFramesTask

Description

This task flags the unreliable signals at the beginning of a chopper plateau.

(A plateau is the chopper on or off or calibration source position inclusive the transition!) Therefore it calculates the differences of the median of a plateau to the single readouts and compares with the maximum tolerance read from the cal file chopJitterThreshold.

Can also be called using the name flagChopMoveFrames!

API Summary

Jython Syntax
`Frames outFrame = cleanPlateauFrames(inFrame [, dmcHead ] [,redundant][, calTree]`
`[, chopJitterThreshold] [, chopperAngle]`
`[, chopperAngleRedundant]`
`[,qualityContext] [, copy])`
`Frames outFrame = flagChopMoveFrames(inFrame [, dmcHead ] [,redundant][, calTree]`
`[, chopJitterThreshold] [, chopperAngle]`
`[, chopperAngleRedundant]`
`[,qualityContext] [, copy])`

Properties
`Frames inFrame [INPUT, Mandatory, default=no default value]`
`PacsDmcProduct dmcHead [INPUT, Optional, default=None]`
`Integer redundant [INPUT, Optional, default=0]`
`PacsCal calTree [INPUT, Optional, default=None]`
`ChopperJitterThreshold chopperJitterThreshold [INPUT, Optional, default=None]`
`ChopperAngle chopperAngle [INPUT, Optional, default=None]`
`ChopperAngleRedundant chopperAngleRedundant [INPUT, Optional, default=None]`
`Integer copy [INPUT, Optional, default=0]`
`QualityContext qualityContext [INOUT, Optional, default=0]`
`Frames outFrame [OUTPUT, Mandatory, default=no default value]`

API details

Properties

Frames inFrame [INPUT, Mandatory, default=no default value]

input Frames object

PacsDmcProduct dmcHead [INPUT, Optional, default=None]

full resolution decmec header

Integer redundant [INPUT, Optional, default=0]

if 0: nominal field plate calibration is used (FPI), if 1: redundant one is used (FPII)

PacsCal calTree [INPUT, Optional, default=None]

CalibrationTree for calfile access

ChopperJitterThreshold chopperJitterThreshold [INPUT, Optional, default=None]

chopper position jitter threshold for sky and calibration source measurements

ChopperAngle chopperAngle [INPUT, Optional, default=None]

chopper calibration readouts to fpu angle

ChopperAngleRedundant chopperAngleRedundant [INPUT, Optional, default=None]

chopper calibration for redundant FP readouts to fpu angle

Integer copy [INPUT, Optional, default=0]

indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

QualityContext qualityContext [INOUT, Optional, default=0]

quality description

Frames outFrame [OUTPUT, Mandatory, default=no default value]

Frames object containing the flags for frames not on the chopper plateaux

1.35. CleanPlateauRampsTask

Full Name:	herschel.pacs.spg.spec.CleanPlateauRampsTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import CleanPlateauRampsTask

Description

Flag the unreliable signals at the beginning of a chopper plateau.

(A plateau is the chopper on or off or calibration source position inclusive the transition!) Therefore it calculates the differences of the median of a plateau to the single readouts and compares with the maximum tolerance read from the cal file chopJitterThreshold.

Can also be called using the name flagChopMoveRamps!

API Summary

Jython Syntax

`Ramps outRamp = cleanPlateauRamps(inRamp [, dmcHead ][,redundant] [, calTree]`
`[, chopJitterThreshold] [, chopperAngle]`
`[, chopperAngleRedundant]`
`[,qualityContext] [, copy])`

`Ramps outRamp = flagChopMoveRamps(inRamp [, dmcHead ][,redundant] [, calTree]`
`[, chopJitterThreshold] [, chopperAngle]`
`[, chopperAngleRedundant]`
`[,qualityContext] [, copy])`

Properties
`Ramps` **`inRamp`** `[INPUT, Mandatory, default=no default value]`
`PacsDmcProduct` **`dmcHead`** `[INPUT, Optional, default=null]`
`Integer` **`redundant`** `[INPUT, Optional, default=0]`
`PacsCal` **`calTree`** `[INPUT, Optional, default=null]`
`ChopperJitterThreshold` **`chopperJitterThreshold`** `[INPUT, Optional, default=null]`
`ChopperAngle` **`chopperAngle`** `[INPUT, Optional, default=null]`
`ChopperAngleRedundant` **`chopperAngleRedundant`** `[INPUT, Optional, default=null]`
`Integer` **`copy`** `[INPUT, Optional, default=0]`
`QualityContext` **`qualityContext`** `[INOUT, Optional, default=0]`
`Ramps` **`outRamp`** `[OUTPUT, Mandatory, default=no default value]`

API details

Properties

Ramps inRamp [INPUT, Mandatory, default=no default value]

input Ramps object

PacsDmcProduct dmcHead [INPUT, Optional, default=null]

full resolution decmec header

Integer redundant [INPUT, Optional, default=0]

redundant if 0: nominal field plate calibration is used (FPI), if 1: redundant one is used (FPII)

PacsCal calTree [INPUT, Optional, default=null]

CalibrationTree for calfile access

ChopperJitterThreshold chopperJitterThreshold [INPUT, Optional, default=null]

chopper position jitter threshold for sky and calibration source measurements

ChopperAngle chopperAngle [INPUT, Optional, default=null]

chopper calibration readouts to fpu angle

ChopperAngleRedundant chopperAngleRedundant [INPUT, Optional, default=null]

chopper calibration for redundant FP readouts to fpu angle

Integer copy [INPUT, Optional, default=0]

indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

QualityContext qualityContext [INOUT, Optional, default=0]

quality control description

Ramps outRamp [OUTPUT, Mandatory, default=no default value]

Ramps object containing the flags for frames not on the chopper plateaux

1.36. CompareRawWithReducedDataFramesTask

Full Name:	herschel.pacs.spg.spec.CompareRawWithReducedDataFramesTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import CompareRawWithReducedDataFramesTask

Description

quality check of onboard (SPU) reduction using the 3 rotating raw ramps (at the same reset intervals),

sets flags if on-ground fit of this raw ramps yields deviating results compared with the slope fitted Frames. Up to now deviation means slope deviation of more than 3.5 digits which is an empirical found value and should be due to rounding accuracy. If a deviation is found, the mask "OBSWERR" will be set to true at the specific pixel and reset interval.

API Summary

Jython Syntax

```
Frames outFrames = compareRawWithReducedDataFrames(Ramps rawRamps,
Frames fitFrames [, int copy = <copy>])
```

Properties
TRamps **rawRamps** [INPUT, MANDATORY, default=NO default value]
Frames **fitFrames** [INPUT, MANDATORY, default=NO default value]
Integer **copy** [INPUT, Optional, default=default value: 0]
Frames **outFrames** [OUTPUT, MANDATORY, default=NO default value]

API details

Properties

TRamps rawRamps [INPUT, MANDATORY, default=NO default value]

input TRamps object raw ramps of 3 rotating pixels stored in TRamps datasets

Frames fitFrames [INPUT, MANDATORY, default=NO default value]

onboard least squares fitted signals stored in a Frames object

Integer copy [INPUT, Optional, default=default value: 0]

indicates if the new fitFrames should be copied and returned (copy =1) or if the old fitFrames is changed (copy = 0)

Frames outFrames [OUTPUT, MANDATORY, default=NO default value]

Depending on copy keyword a copy of the fitFrames class or an updated Frames class with according flags

History

- 2009-03-25 - JS: 1.0 converted from jython to java due to performance reasons

1.37. CompareRawWithReducedDataFramesTask

Full Name:	herschel.pacs.toolboxes.spg.CompareRawWithReducedDataFramesTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spg import CompareRawWithReducedDataFramesTask

Description

quality check of onboard (SPU) reduction using the 3 rotating raw ramps (at the same reset intervals),

sets flags if on-ground fit of this raw ramps yields deviating results compared with the slope fitted Frames. Up to now deviation means slope deviation of more than 3 digits/s which is an empirical found value and should be due to rounding accuracy. If a deviation is found, the whole array will be masked for that reset.

API Summary

Jython Syntax
`outFrames = compareRawWithReducedDataFrames(Ramps rawRamps, Frames fitFrames [, int copy = <copy>])`

Properties
`TRamps rawRamps [INPUT, MANDATORY, default=NO default value]`
`Frames outFrames [OUTPUT, MANDATORY, default=NO default value]`
`Frames fitFrames [INPUT, MANDATORY, default=NO default value]`
`Integer copy [INPUT, Optional, default=default value: 0]`

API details

Properties

`TRamps rawRamps [INPUT, MANDATORY, default=NO default value]`
raw ramps of 3 rotating pixels stored in TRamps datasets

`Frames outFrames [OUTPUT, MANDATORY, default=NO default value]`
Depending on copy keyword a copy of the Frames class or an updated Frames class with flags

`Frames fitFrames [INPUT, MANDATORY, default=NO default value]`
onboard least square fitted signals stored in Frames class

`Integer copy [INPUT, Optional, default=default value: 0]`
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

History

- 1.0 24-Aug-2007 JS first version

- 1.1 21-May-2008 JS improved version, only rounding errors considered now

1.38. CompareRawWithReducedDataRampsTask

Full Name:	herschel.pacs.spg.spec.CompareRawWithReducedDataRampsTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import CompareRawWithReducedDataRampsTask

Description

quality check of onboard (SPU) reduction using the 3 rotating raw ramps (at the same reset intervals),

sets flags if ramp averaging of this raw ramps yields deviating results compared with the reduced Ramps values. Deviating means that the differences of the Integer converted averaged values of the raw ramps and the subramp averaged ramps deviate more than 1 which could be due to rounding accuracy. If a deviation is found, the mask "OBSWERR" will be set to true at the specific pixel and reset interval.

API Summary

Jython Syntax

```
Ramps outRamps = compareRawWithReducedDataRamps(Ramps rawRamps,
Ramps reducedRamps [, int copy = <copy>])
```

Properties

TRamps **rawRamps** [INPUT, MANDATORY, default=NO default value]
Ramps **reducedRamps** [INPUT, MANDATORY, default=NO default value]
Integer **copy** [INPUT, Optional, default=default value: 0]
Ramps **outRamps** [OUTPUT, MANDATORY, default=NO default value]

API details

Properties

TRamps rawRamps [INPUT, MANDATORY, default=NO default value]

input TRamps object raw ramps of 3 rotating pixels stored in TRamps datasets

Ramps reducedRamps [INPUT, MANDATORY, default=NO default value]

onboard reduced, averaged ramps stored in ARamps datasets

Integer copy [INPUT, Optional, default=default value: 0]

indicates if the new ramps should be copied and returned (copy =1) or if the old ramp is changed (copy = 0)

Ramps outRamps [OUTPUT, MANDATORY, default=NO default value]

Depending on copy keyword a copy of the Ramps class or an updated Ramps class with according flags

History

- 2009-03-20 - JS: 1.0: converted from jython to java due to performance reasons

1.39. CompareRawWithReducedDataRampsTask

Full Name:	herschel.pacs.toolboxes.spg.CompareRawWithReducedDataRampsTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spg import CompareRawWithReducedDataRampsTask

Description

quality check of onboard (SPU) reduction using the 3 rotating raw ramps (at the same reset intervals),

sets flags if ramp averaging of this raw ramps yields deviating results compared with the reduced Ramps values. Deviating means that the differences of the Integer converted averaged values of the raw ramps and the subramp averaged ramps deviate more than 1 which could be due to rounding accuracy. If a deviation is found, the whole array will be masked for the according reset interval.

API Summary

Jython Syntax
`outRamps = compareRawWithReducedDataRamps(Ramps rawRamps, Ramps reducedRamps [, int copy = <copy>])`

Properties
`Ramps rawRamps [INPUT, MANDATORY, default=NO default value]`
`Ramps reducedRamps [INPUT, MANDATORY, default=NO default value]`
`Ramps outRamps [OUTPUT, MANDATORY, default=NO default value]`
`Integer copy [INPUT, Optional, default=default value: 0]`

API details

Properties

Ramps rawRamps [INPUT, MANDATORY, default=NO default value]
raw ramps of 3 rotating pixels stored in TRamps datasets

Ramps reducedRamps [INPUT, MANDATORY, default=NO default value]
onboard reduced, averaged ramps stored in ARamps datasets

Ramps outRamps [OUTPUT, MANDATORY, default=NO default value]
Depending on copy keyword a copy of the Ramps class or an updated Ramps class with according flags

Integer copy [INPUT, Optional, default=default value: 0]
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

History

- 1.1 22-Aug-2007 JS first version

- 1.2 24-Aug-2007 JS bug fix
- 1.3 21-May-2008 JS improved version, only rounding errors considered now

1.40. ComputeDmcRampsTask

Full Name:	herschel.pacs.spg.level0.ComputeDmcRampsTask
Type:	Java Task -
Import:	from herschel.pacs.spg.level0 import ComputeDmcRampsTask

Description

This task reduces the size of the Ramps status entries.

Right after population, Ramps objects contain a full resolution decmec header as status.

On the other hand, Ramps objects can contain onboard reduced signal values. As a consequence, the resolution of the status and the signal are not the same.

This task analyses the algorithm keyword from the Ramps metadata and applies the corresponding reduction algorithm also to the Ramps status object.

Example

Example 1: Reduce the status of an onboard averaged Ramps object, copy the orignal Ramps object

```
outramps = computeDmcRamps(inramps, copy = True)<br>
<br>
edit the rules for status reduction<br>
drules = computeDmcRamps.getDefaultRules()<br>
#print the rule for the CRECR status value<br>
print drules.get("CRECR") #1<br>
#remove CRECR from the status<br>
drules.put("CRECR", 0)<br>
#get the VLD entry as 1d instead of reduced 2d<br>
drules.put("VLD", 1)<br>
#apply the new rules<br>
outramps = computeDmcRamps(inramps, copy = True, rules = drules)<br>
<br><br>
```

API Summary

Jython Syntax

```
from herschel.pacs.spg import ComputeDmcRampsTask<br>
computeDmcRamps = ComputeDmcRampsTask()<br>
<y>outramps = computeDmcRamps(inramps [, copy, rules,
algorithm])<br>
<br>
```

Properties

the Ramps object with the full resolution status **inramps** [INPUT, MANDATORY, default=no default value]
type **rules** [A java.util.HasMap with the rules for status reduction. The HashMap key is a String with, MANDATORY, default=no default value]
type **algorithm** [an instance of herschel.pacs.spu.SpuAlgorithm., MANDATORY, default=no default value]

Properties
type **copy** [true: return a copy of the input Ramps object with the modified status, MANDATORY, default=no default value]
the returned Ramps object with the reduced status. **outramps** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

the Ramps object with the full resolution status inramps [INPUT, MANDATORY, default=no default value]

type rules [A java.util.HasMap with the rules for status reduction. The HashMap key is a String with, MANDATORY, default=no default value]

the name of the status entry, the value is an integer with three possible values:

0: remove the status entry

1: take the first entry for every reset interval and return the status as a 1d column using these values

2: apply the reduction algorithm to the status values (and thus return a 2d reduced status entry)

The default set of rules can be obtained, analysed and edited with the method getDefaultRules() or

by executing print ComputeDmcRampsTask() (in jython)

type algorithm [an instance of herschel.pacs.spu.SpuAlgorithm., MANDATORY, default=no default value]

By default, the right Algorithm is chosen from the Ramps meta data. The user supplied algorithm can overwrite the default algorithm

type copy [true: return a copy of the input Ramps object with the modified status, MANDATORY, default=no default value]

false (default): overwrite the status object of the input Ramps

the returned Ramps object with the reduced status. outramps [OUTPUT, MANDATORY, default=no default value]

1.41. ComputeDpuTimeCorrectionTask

Full Name:	herschel.pacs.toolboxes.common.ComputeDpuTimeCorrectionTask
Type:	Java Task -
Import:	from herschel.pacs.toolboxes.common import ComputeDpuTimeCorrectionTask

Description

Compute the runaway time of the "DPU Bus protokoll".

During ILT we discovered that the "DPU Time" of PACS Packets were "running away from each other"

The CDMS simulator send its time to the DPU frequently. But it seems that the Bus inbetween

has its own time.

This tool takes every x seconds the time from a CDMS simulator and the consecutive PACS science packets.

SciencePackets are necessary to guarantee the accuracy

The program takes the valus and generates a Calibration Table for Time correction.

1.42. CondenseBoxcar

Full Name:	herschel.pacs.share.numeric.CondenseBoxcar
Type:	Java Class -
Import:	from herschel.pacs.share.numeric import CondenseBoxcar

Description

Default mode : Condense array x along dimension d applying function f using a truncation size of a boxcar filter

API Summary

Jython Syntax
`outArray = CondenseBoxcar ( alongDimension, chunkSize, function) (inArray)`

Properties
`float or double array` **`inArray`** `[INPUT, MANDATORY, default=p]`
`integer` **`alongDimension`** `[INPUT, MANDATORY, default=p]`
`int` **`chunkSize`** `[INPUT, MANDATORY, default=p]`
`ArrayReductor|ArrayToNumber` **`function`** `[INPUT, MANDATORY, default=p]`
`float or double array` **`outArray`** `[OUTPUT, MANDATORY, default=no default value]`

API details

Properties

`float or double array inArray [INPUT, MANDATORY, default=p]`

Input array of n dimensions

`integer alongDimension [INPUT, MANDATORY, default=p]`

Dimension in which the function will be applied

`int chunkSize [INPUT, MANDATORY, default=p]`

Size of chunks to process

`ArrayReductor|ArrayToNumber function [INPUT, MANDATORY, default=p]`

Function to apply

`float or double array outArray [OUTPUT, MANDATORY, default=no default value]`

Return an array where the function was applied on data chunks in dimension

The center of the box is always the actual index. in case of an "even" box the value before the theorhetical center

1.43. ConvertChopper2AngleTask

Full Name:	herschel.pacs.spg.ConvertChopper2AngleTask
Type:	Jython Task -
Import:	from herschel.pacs.spg import ConvertChopper2AngleTask

Description

Jython task that calculates the chopper position angle with respect to the optical axis of

the focal plane unit in degrees and adds the Status parameter CHOPFPUANGLE.

Additionally, the angle on sky in arcmin is added as Status parameter (CHOPSKYANGLE).

Both values are computed by reading the Status DEC/MEC parameter "CPR" in position readouts, converting them using the corresponding calfiles ChopperAngle and ChopperSkyAngle and then populating the new Status words in the returned Frames object. The same can be done for the redundant chopper controller!

API Summary

Jython Syntax
`Frames outFrames = convertChopper2Angle(Frames inFrames [, redundant = 0][, CalibrationTree calTree][, ChopperSkyAngle chopperSkyAngle][, int copy = 0] )`

Properties
`Frames inFrames [INOUT, MANDATORY, default=NO default value]`
`int redundant [INPUT, Optional, default=default value: 0]`
`CalibrationTree calTree [INPUT, Optional, default=default value: None]`
`ChopperSkyAngle chopperSkyAngle [INPUT, Optional, default=default value: None]`
`ChopperAngle chopperAngle [INPUT, Optional, default=default value: None]`
`ChopperAngleRedundant chopperAngleRedundant [INPUT, Optional, default=default value: None]`
`Integer copy [INPUT, Optional, default=default value: 0]`
`Frames outFrame [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

Frames inFrames [INOUT, MANDATORY, default=NO default value]
input Frames object

int redundant [INPUT, Optional, default=default value: 0]
if 0: nominal field plate calibration is used (FPI), if 1: redundant one is used (FPII)

CalibrationTree calTree [INPUT, Optional, default=default value: None]

conversion factor from chopper fpu angle to sky angle cal file access

ChopperSkyAngle chopperSkyAngle [INPUT, Optional, default=default value: None]

conversion factor from chopper fpu angle to sky angle cal file access

ChopperAngle chopperAngle [INPUT, Optional, default=default value: None]

chopper calibration readouts to fpu angle cal file access

ChopperAngleRedundant chopperAngleRedundant [INPUT, Optional, default=default value: None]

chopper calibration for redundant FP readouts to fpu angle cal file access

Integer copy [INPUT, Optional, default=default value: 0]

indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

Frames outFrame [OUTPUT, MANDATORY, default=NO default value]

returned Frames object containing the additional status parameters

History

- 1.0 20051208 JS initial version - prototype
- 1.1 20061107 JS changed to use more precise chopPos2FpuAnglePol
- 1.2 20061129 JS additional redundant parameter introduced
- 1.3 20070314 JS parameter zerooffset introduced
- 1.4 20070501 EkW correct addFloatStatus
- 1.5 20071121 JS conversion to new cal framework
- 1.6 20080205 JS adopt new interfaces
- 1.7 20080221 JS shift optional parameter redundant in front of cal parameters
- 1.8 20081112 JS remove optional parameter zeroOffset, now in cal file ChopperSkyAngle
- 1.9 20081211 JS implementation of SPR 1230: subtraction of zeroOffset

1.44. convertChopperReadoutSteps

Full Name:	herschel.pacs.toolboxes.common.convertChopperReadoutSteps
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import convertChopperReadoutSteps

Description

Jython function that calculates the given chopper step readouts in angle units (arcmin) at a given setpoint deflection

using the cal file ChopperAngle, because the thresholds depend non-linearly on the chopper deflection angle. The function uses the derivation of the chopper calibration polynomial at the given setpoint.

API Summary

Jython Syntax
`Double1d stepAngle = convertChopperReadoutSteps(Int1d setpoint, Double readoutStep [, redundant = 0] [, CalibrationTree calTree][,ChopperAngle chopperAngle][,ChopperAngleRedundant chopperAngleRedundant]` `[,String model][,FineTime time])`

Properties
`int` **`setpoint`** `[INPUT, MANDATORY, default=NO default value]`
`Double` **`readoutStep`** `[INPUT, MANDATORY, default=NO default value]`
`int` **`redundant`** `[INPUT, Optional, default=default value: 0]`
`CalibrationTree` **`calTree`** `[INPUT, Optional, default=default value: None]`
`ChopperAngle` **`chopperAngle`** `[INPUT, Optional, default=default value: None]`
`ChopperAngleRedundant` **`chopperAngleRedundant`** `[INPUT, Optional, default=default value: None]`
`String` **`model`** `[INPUT, Optional, default=default value: "FM"]`
`FineTime` **`time`** `[INPUT, Optional, default=default value: present date]`
`Double` **`stepAngle`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`int setpoint [INPUT, MANDATORY, default=NO default value]`
input commanded chopper deflection in DecMec readouts

`Double readoutStep [INPUT, MANDATORY, default=NO default value]`
input chopper deflection angle step in readouts

int redundant [INPUT, Optional, default=default value: 0]
if 0: nominal field plate calibration is used (FPI), if 1: redundant one is used (FPII)

CalibrationTree calTree [INPUT, Optional, default=default value: None]
cal file access

ChopperAngle chopperAngle [INPUT, Optional, default=default value: None]
chopper calibration readouts to fpu angle cal file access

ChopperAngleRedundant chopperAngleRedundant [INPUT, Optional, default=default value: None]
chopper calibration for redundant FP readouts to fpu angle cal file access

String model [INPUT, Optional, default=default value: "FM"]
model name FM, FS or QM

FineTime time [INPUT, Optional, default=default value: present date]
time of measurement

Double stepAngle [OUTPUT, MANDATORY, default=NO default value]
chopper readout step converted to deflection specific angle units (arcmins)

History

- 1.0 20060420 JS initial version
- 1.1 20060509 JS changed name from convertThreshold to convertChopper*Steps
- to generalize for arbitrary chopper steps
- 1.2 20060511 JS imports updated
- 1.3 20060512 JS imports updated
- 1.4 20060620 JS input for chopPos2FpuAnglePol converted to Double1d
- 1.5 20060714 JS bug fix
- 1.6 20061128 JS changed to give arrays as In-and Output
- 1.7 20071122 JS adaption to new cal framework
- 1.8 20080222 JS adapt again to new cal framework

1.45. ConvertSignal2StandardCapTask

Full Name:	herschel.pacs.spg.spec.ConvertSignal2StandardCapTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.spec import ConvertSignal2StandardCapTask

Description

Jython spectrometer related task that reads the capacitance ratios cal file

and scales all signals to the lowest available integration capacitance which is referred to as the standard capacitance. This is done to be able to compare signals recorded using different integration capacitances

API Summary

Jython Syntax
`Frames outFrame = convertSignal2StandardCap(Frames inFrame [,calTree][,capacitanceRatios][, int copy = <copy>])`

Properties
`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
`PacsCal calTree [INPUT, Optional, default=default value: None]`
`CapacitanceRatios capacitanceRatios [INPUT, Optional, default=default value: None]`
`Integer copy [INPUT, Optional, default=default value: 0]`
`Frames outFrame [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

Frames inFrame [INPUT, MANDATORY, default=NO default value]
input Frames object

PacsCal calTree [INPUT, Optional, default=default value: None]
cal file access

CapacitanceRatios capacitanceRatios [INPUT, Optional, default=default value: None]
cal file access

Integer copy [INPUT, Optional, default=default value: 0]
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

Frames outFrame [OUTPUT, MANDATORY, default=NO default value]
returned Frames containing the converted signals [V/s]

History

- 1.0 20060530 JS initial prototype version
- 1.1 20071115 EkW adopt to new calibration framework
- 1.2 20071115 JS interface of getEffectiveCapacitances adapted to new cal framework
- 1.3 20080221 JS adapt again to new cal framework
- 1.4 20080306 JS adapted according to proposals in email of PR

1.46. ConvXyStage2PointingTask

Full Name:	herschel.pacs.toolboxes.common.ConvXyStage2PointingTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import ConvXyStage2PointingTask
Category:	pacs/toolboxes/spg

Description

Add the XyStage "Pointing Information" ("Stage_X" and "Stage_Y" in mm) to Frames class

Examples

Example 1: assign XY stage coordinates with Interpolation

```
frames = convXyStage2Pointing(frames)
```

Example 2: assign XY stage coordinates without Interpolation

```
frames = convXyStage2Pointing(frames, noInter= True)
```

API Summary

Jython Syntax

```
frames = convXyStage2Pointing(frames, seq , noInter=noInter, copy
= copy)
```

Properties

Frames **frames** [IN/OUT, MANDATORY, default=NO default value]
String **seq** [INPUT, MANDATORY, default=No default value]
boolean **noInter** [INPUT, OPTIONAL, default=default value : False]
Integer **copy** [INPUT, Optional, default=default value: 0]

Limitations

No Limitation

Miscellaneous

Only used for ILT with XY Stage

API details

Properties

Frames frames [IN/OUT, MANDATORY, default=NO default value]

Frames input (output, if copy = 1 is given)

String seq [INPUT, MANDATORY, default=No default value]
PacketSequence containing the XyTelemetry Packets belonging to the period of the Frames class

boolean noInter [INPUT, OPTIONAL, default=default value : False]
If true : NO interpolation is done between the XY HK packets

Integer copy [INPUT, Optional, default=default value: 0]
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

See also

- ???

History

- 06-Apr-2006 EkW Initial version of this Task
- 16-Aug-2006 AC Add parameter to Frame Status
- 14-Nov-2006 EkW Use mergeFramesHk
- 14-Dec-2006 EkW add noInt keyword
- 23-May-2007 EkW Correction of Time access SPR 723
- 01-Feb-2008 EkW In case of "late" XY stage packet we extrapolate backward
- 21-Feb-2008 EkW Documentation
- 01-Apr-2008 EkW Change x -> -y and y -> -x as proposed by Dave and Dieter
- 24-Apr-2008 EkW SPR 856

1.47. CorrectDpuTimeFramesTask

Full Name:	herschel.pacs.toolboxes.spg.CorrectDpuTimeFramesTask
Type:	Java Task -
Import:	from herschel.pacs.toolboxes.spg import CorrectDpuTimeFramesTask

Description

Correct Frames DecMec time for the runaway time of the "DPU Bus protokoll".

During ILT we discovered that the "DPU Time" of PACS Packets were "running away from each other"

The CDMS simulator send its time to the DPU frequently. But it seems that the Bus inbetween

has its own time.

This tool takes every x seconds the time from a CDMS simulator and the consecutive PACS science packets.

SciencePackets are necessary to guarantee the accuracy

The program corrects the FramesDecMec time accordingly

API Summary

Jython Syntax
`Frames framesOut = correctDpuTimeFrames(Frames frames, PacketSequence seq [,Integer copy=copy]) <p>`

Properties
`Frames frames [INPUT, MANDATORY, default=NO default value]`
`PacketSequence seq [INPUT, MANDATORY, default=NO default value]`
`Integer copy [INPUT, OPTIONAL, default=0]`

API details

Properties

`Frames frames [INPUT, MANDATORY, default=NO default value]`
input Frames object

`PacketSequence seq [INPUT, MANDATORY, default=NO default value]`
corresponding PacketSequence

`Integer copy [INPUT, OPTIONAL, default=0]`
copy : 0 (default) - give back a reference, 1: give back a copy

1.48. CorrectRaDec4SsoTask

Full Name:	herschel.pacs.toolboxes.spg.CorrectRaDec4SsoTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spg import CorrectRaDec4SsoTask

Description

Move SSO target to a fixed position in sky. This is needed for mapping.

API Summary

Jython Syntax
`frames = correctRaDec4Sso(frames, horizons [copy])`

Properties
`String` **`copy`** `[INPUT, OPTIONAL, default=default value : 0]`
`frames class with corrected Ra and dec valus in the Status` **`frames`** `[OUT, MANDATORY, default=no default value]`

API details

Properties

`String copy [INPUT, OPTIONAL, default=default value : 0]`
Copy the input (0) and generate new output or overwrite the input (1)

`frames class with corrected Ra and dec valus in the Status frames [OUT, MANDATORY, default=no default value]`

1.49. creBiasLoopOptimum

Full Name:	herschel.pacs.scripts.iascripts.creBiasLoopOptimum
Type:	Jython Task -
Import:	from herschel.pacs.scripts.iascripts import creBiasLoopOptimum

Description

Calculate optimum bias settings for PACS Ge:Ga detectors

API Summary

Jython Syntax
`Execute step by step in the command line`

Property
`type frames [INPUT, MANDATORY, default=no default value]`

Miscellaneous

This script needs to be revised for FM/ILT!

API details

Property

`type frames [INPUT, MANDATORY, default=no default value]`
No

History

- 0.1 21-Oct-2004 RV Initial version of this definition
- 0.2 25-Feb-2005
- 0.7 26-May-2006

1.50. creDynRangeAnalysisExec

Full Name:	herschel.pacs.scripts.iascripts.creDynRangeAnalysisExec
Type:	Jython Task -
Import:	from herschel.pacs.scripts.iascripts import creDynRangeAnalysisExec

Description

Master file to evaluate dynamic range for CQM detectors

API Summary

Jython Syntax
`Run in the command line`

Property
`type frames [INPUT, MANDATORY, default=no default value]`

Miscellaneous

Applicable only for CQM data! This script needs to be revised for EQM/IMT and FM/ILT

API details

Property

`type frames [INPUT, MANDATORY, default=no default value]`
No

History

- 0.1 21-Oct-2004 RV Initial version of this definition
- 0.2 25-Feb-2005
- 0.7 26-May-2006

1.51. creDynRangeAnalysis

Full Name:	herschel.pacs.scripts.iascripts.creDynRangeAnalysis
Type:	Jython Task -
Import:	from herschel.pacs.scripts.iascripts import creDynRangeAnalysis

Description

Functionality toolbox to evaluate dynamic range for CQM detectors

API Summary

Jython Syntax
`dynamicRangeAnalysis_v07(dfs, channel, refPixFind, timeSlip)`
`dynamicRangeForArray_v02(dfSeq,chn,subRampsToDiscardIn)`
`badPixelMaskEqm(channel)`

Property
`type frames [INPUT, MANDATORY, default=no default value]`

Miscellaneous

Applicable only for CQM data! This script needs to be revised for EQM/IMT and FM/ILT

API details

Property

`type frames [INPUT, MANDATORY, default=no default value]`
No

History

- 0.1 21-Oct-2004 RV Initial version of this definition
- 0.2 25-Feb-2005
- 0.7 26-May-2006

1.52. creDynRangeAnalysisRead

Full Name:	herschel.pacs.scripts.iascripts.creDynRangeAnalysisRead
Type:	Jython Task -
Import:	from herschel.pacs.scripts.iascripts import creDynRangeAnalysisRead

Description

Create plots for dynamic range analysis of the CQM detectors

API Summary

Jython Syntax
`Execute in the Jide command line`

Property
`type frames [INPUT, MANDATORY, default=no default value]`

Miscellaneous

Applicable only for CQM data! This script needs to be revised for EQM/IMT and FM/ILT

API details

Property

`type frames [INPUT, MANDATORY, default=no default value]`
No

History

- 0.1 21-Oct-2004 RV Initial version of this definition
- 0.2 25-Feb-2005
- 0.7 26-May-2006

1.53. CubeBuilder

Full Name:	herschel.pacs.spg.CubeBuilder
Type:	Java Class -
Import:	from herschel.pacs.spg import CubeBuilder

Description

Help taken fromn CubeBuilderTask

API Summary

Properties
Frames **inFrames** [INPUT, MANDATORY, default=NO default value]
Integer **module** [INPUT, OPTIONAL, default=default value : -1]
Integer **startFrame** [INPUT, OPTIONAL, default=default value : -1]
Integer **endFrame** [INPUT, OPTIONAL, default=default value : -1]
Integer **averageWaves** [INPUT, OPTIONAL, default=default value : 0]
Integer **copy** [INPUT, OPTIONAL, default=default value : 1]
Frames **frame** [OUTPUT, MANDATORY, default=NO default value]

API details

Properties

Frames inFrames [INPUT, MANDATORY, default=NO default value]

input Frames object

Integer module [INPUT, OPTIONAL, default=default value : -1]

select row (module) number

Integer startFrame [INPUT, OPTIONAL, default=default value : -1]

select starting frame in timeline

Integer endFrame [INPUT, OPTIONAL, default=default value : -1]

select finishing frame in timeline

Integer averageWaves [INPUT, OPTIONAL, default=default value : 0]

If 1, the time cube is rearranged to be 5x5 instead of 18x25. all the wavelengths, modules between 1 and 16, are averaged.

Integer copy [INPUT, OPTIONAL, default=default value : 1]

Copy the input (1) and generate new output or overwrite the input (0)

Frames frame [OUTPUT, MANDATORY, default=NO default value]

returned frame with 5x5 cube representing integral field view.

1.54. CWT

Full Name:	herschel.pacs.spg.phot.deglitching.WTMML.util.CWT
Type:	Java Class -
Import:	from herschel.pacs.spg.phot.deglitching.WTMML.util import CWT

Description

This class performs a continuous wavelet transform on a signal

using a Mexican Hat wavelet (filter). The coefficients W(a,b) are computed by a convolution of the signal with the * filter function. The decomposition is performed on a given number of octaves **noct** using a given number of voices **nvoice**. The voice number **v** and the octave number **o** are related with the scale decomposition **a** by :

```
a = amin 2 noct + v/nvoice
```

History

- 24/04/2006 : first java version
- october 2006 : add noiseEstimation method

1.55. DataratesTask

Full Name:	herschel.pacs.toolboxes.binstruct.DataratesTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.binstruct import DataratesTask

Description

This function calculates datarates for a packet sequence.

API Summary

Jython Syntax
`TableDataset rates = dataRates(PacketSequence seq[, integer binsize=<binsize>])`

Properties
`PacketSequence` **`seq`** `[INPUT, MANDATORY, default=NO default value]`
`integer` **`binsize`** `[INPUT, OPTIONAL, default=1]`
`TableDataset` **`rates`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`PacketSequence seq [INPUT, MANDATORY, default=NO default value]`
a packet sequence for which to calculate the datarate

`integer binsize [INPUT, OPTIONAL, default=1]`
the size of the bins in seconds

`TableDataset rates [OUTPUT, MANDATORY, default=NO default value]`
a table containing the datarates for each time bin in kbits/s

History

- 28-Sep-2004 RH Initial version of this task
- 28-Jul-2005 JS new header according to jtags framework
- 07-Nov-2005 JS clean up of imports
- 03-Mar-2006 EkW fix initialization

1.56. DeactivateMasksTask

Full Name:	herschel.pacs.spg.DeactivateMasksTask
Type:	Java Task -
Import:	from herschel.pacs.spg import DeactivateMasksTask

Description

Task that set the status of a submitted masklist to inactive.

API Summary

Jython Syntax
`out = activateMasks(in ,["BLINDPIXELS", "SATURATION"])`

Properties
PhotRaw **in : PacsProduct (Frames** [Ramps, PacsCube) that contains masks, default=no default value]
type **masks: List of masknames. The specified masks will be set to inactive** [INPUT, MANDATORY, default=no default value]
type **exclusive: if True** [the specified Masks will be set as activate commands. All others will be set to the complementary state., MANDATORY, default=no default value]
type **out : the same PacsProduct as in but with changed mask status** [INPUT, MANDATORY, default=no default value]

API details

Properties

PhotRaw in : PacsProduct (Frames [Ramps, PacsCube) that contains masks, default=no default value]

type masks: List of masknames. The specified masks will be set to inactive [INPUT, MANDATORY, default=no default value]

type exclusive: if True [the specified Masks will be set as activate commands. All others will be set to the complementary state., MANDATORY, default=no default value]

if False, the state of other masks is not touched. Default value is False.

type out : the same PacsProduct as in but with changed mask status [INPUT, MANDATORY, default=no default value]

1.57. DecodeBbidTask

Full Name:	herschel.pacs.toolboxes.common.DecodeBbidTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import DecodeBbidTask

Description

Extract the 14 bits BBID from Bbid number : 2Bits : instrument, 14Bits : BBType, 16 Bits: BB Execution Counter

API Summary

Jython Syntax
`bbidOut = decodeBbid(bbidIn)`

Properties
`Long1d bbid [INPUT, BBID, default=the complete BBID identifier]`
`OUTPUT unknown [bbidOut, Long1d, default=the extracted 14 Bits BBID]`

API details

Properties

Long1d bbid [INPUT, BBID, default=the complete BBID identifier]

OUTPUT unknown [bbidOut, Long1d, default=the extracted 14 Bits BBID]
.

History

- 1.0 30-Nov-2006 EkW The first version

1.58. DecodeBolStatusTask

Full Name:	herschel.pacs.toolboxes.phot.DecodeBolStatusTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.phot import DecodeBolStatusTask

Description

Decode the BOLC Status word

Just applies Bit Operations to decode the Bitpattern.

API Summary

Jython Syntax
`TableDataset statusTable = decodeBolStatus(Int1d stat)`

Properties
`Int1d stat [INPUT, MANDATORY, default=NO default value]`
`TableDataset statusTable [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Int1d stat [INPUT, MANDATORY, default=NO default value]`
Bol status words as coming from DecMec

`TableDataset statusTable [OUTPUT, MANDATORY, default=NO default value]`
TableDataset containing the decoded values Decoded Bol Status as TableDataset

History

- 1.0 10-Jan-2005 EkW Initial version of this task
- 1.1 04-Aug-2005 JS header adopted to jtags syntax
- 1.2 07-Nov-2005 JS clean up of imports

1.59. DecodeCapacitanceTask

Full Name:	herschel.pacs.spg.level0.DecodeCapacitanceTask
Type:	Java Task -
Import:	from herschel.pacs.spg.level0 import DecodeCapacitanceTask
Category:	Task/PACS Task

Description

This task decodes CRECR status and put the capacitance in pF either into

the status of Frames/Ramps or in the meta header

Use it as follows:

ramps = decodeCapacitance(ramps [, calTree = calTree][, effectiveCapacitance = efCap][, meta = meta]) frames = decodeCapacitance(frames [, calTree = calTree][, effectiveCapacitance = efCap][, meta = meta])

Examples

Example 1: from java:

```
The best call from java is not the task, but the application:
frames = DecodeCapacitance.decodeCapacitance(frames) or
ramps = DecodeCapacitance.decodeCapacitance(ramps)
```

Example 2: from jython:

```
from herschel.pacs.spg import DecodeCapacitanceTask
decodeCapacitance = DecodeCapacitanceTask()
#f = Frames(), r = Ramps()
frames = decodeCapacitance(f) or  frames = decodeCapacitance(frames = f)
ramps = decodeCapacitance(r) or ramps = decodeCapacitance(ramps = r)
Possible is also:
decodeCapacitance(frames = f, ramps = r)
In this case f and r will be modified by the decode capacitance task.
```

API Summary

Jython Syntax

```
ramps = decodeCapacitance(ramps [, calTree = calTree][,
effectiveCapacitance = efCap][, meta = meta])
frames = decodeCapacitance(frames [, calTree = calTree][,
effectiveCapacitance = efCap][, meta = meta])
```

Properties

Frames **frames** [INPUT, MANDATORY, default=no default value]
Ramps **ramps** [INPUT, MANDATORY, default=no default value]
PacsCal **calTree** [INPUT, Optional, default=default value: None]
EffectiveCapacitance **effectiveCapacitance** [INPUT, Optional, default=default value: None]
Integer **meta** [INPUT, Optional, default=default value: 0]

Properties
Frames **frames** [OUTPUT, MANDATORY, default=no default value]
Ramps **ramps** [OUTPUT, MANDATORY, default=no default value]

Limitations

Do NOT use: frames = decodeCapacitance(frames = f, ramps = r) or frames = decodeCapacitance(f, ramps = r) the syntax is possible, but decodeCapacitance will return only the first value which is frames in this case

API details

Properties

Frames frames [INPUT, MANDATORY, default=no default value]
input frames object

Ramps ramps [INPUT, MANDATORY, default=no default value]
input ramps object

PacsCal calTree [INPUT, Optional, default=default value: None]
input calibration tree

EffectiveCapacitance effectiveCapacitance [INPUT, Optional, default=default value: None]
input cal file

Integer meta [INPUT, Optional, default=default value: 0]
indicates if the cap should go into meta (meta =1) or if the it goes into the status (meta = 0)

Frames frames [OUTPUT, MANDATORY, default=no default value]
the updated frames object

Ramps ramps [OUTPUT, MANDATORY, default=no default value]
the updated ramps object

History

- 2009-06-14 - JS: 1.0 Initial version of this Task

1.60. decodeCompAlgo

Full Name:	herschel.pacs.toolboxes.common.decodeCompAlgo
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import decodeCompAlgo

Description

Convert Compression Algorithm information of CompressedEntity Header into human readable String format

API Summary

Jython Syntax
`String algoString = decodeCompAlgo(int algoInt)`

Properties
`int algoInt [INPUT, MANDATORY, default=NO default value]`
`String algoString [OUTPUT, MANDATORY, default=NO default value]`

Miscellaneous

Shall use a calibration file later. Currently information is hardcoded.

API details

Properties

`int algoInt [INPUT, MANDATORY, default=NO default value]`

Compression Algorithm number comming from Compressed Entity header Be aware that this number may not make sense in combination of a without reduction ! OBSW keeps just the last number.

`String algoString [OUTPUT, MANDATORY, default=NO default value]`

Human readable from of the compression Algorithm

See also

- ???

History

- 1.0 01-Apr-2004 EkW Initial version of this task
- 1.1 03-Aug-2005 JS Header adopted to jtags concept
- 1.2 06-Feb-2006 EkW Update documentation

1.61. decodeCompMode

Full Name:	herschel.pacs.toolboxes.common.decodeCompMode
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import decodeCompMode

Description

Convert CompressionMode information of CompressedEntity Header into human readable String format

API Summary

Jython Syntax
`String modeString = decodeCompMode(int modeInt)`

Properties
`int modeInt [INPUT, MANDATORY, default=NO default value]`
`String modeString [OUTPUT, MANDATORY, default=NO default value]`

Miscellaneous

Shall use a calibration file later. Currently information is hardcoded.

API details

Properties

int modeInt [INPUT, MANDATORY, default=NO default value]
Compression Mode number comming from Compressed Entity header

String modeString [OUTPUT, MANDATORY, default=NO default value]
Human readable from of the compression mode

See also

- ???

History

- 1.0 01-Apr-2004 EkW Initial version of this task
- 1.1 03-Aug-2005 JS header adopted to jtags concept

1.62. DecodeLabel

Full Name:	herschel.pacs.spg.level0.DecodeLabel
Type:	Java Class -
Import:	from herschel.pacs.spg.level0 import DecodeLabel

History

- 1.0 18-Aug-2005 EkW Initial version of this Task
- 1.1 07-Nov-2005 JS clean up of imports
- 2.0 24-Jan-2007 MW recoded to Java
- 2.1 08-Feb-2008 MW extended to work also for Ramps
- 2.2 14-Mar-2008 EkW adoption to binstruct and general clean up

1.63. DecodeLabelTask

Full Name:	herschel.pacs.spg.level0.DecodeLabelTask
Type:	Java Task -
Import:	from herschel.pacs.spg.level0 import DecodeLabelTask

Description

This task decomposes the Label and put it into the status of Frames/Ramps

Use it as follows.

Examples

Example 1: from java:

```
The best call from java is not the task, but the application:
frames = DecodeLabel.decodeLabel(frames) or
ramps = DecodeLabel.decodeLabel(ramps
```

Example 2: from jython:

```
from herschel.pacs.spg import DecodeLabelTask
decodeLabel = DecodeLabelTask()
#f = Frames(), r = Ramps()
frames = decodeLabel(f) or  frames = decodeLabel(frames = f)
ramps = decodeLabel(r) or ramps = decodeLabel(ramps = r)
Possible is also:
decodeLabel(frames = f, ramps = r)
In this case f and r will be modified by the decode label task.
```

API Summary

Properties
Frames **frames** [INPUT, Optional, default=None]
Ramps **ramps** [INPUT, Optional, default=None]
Frames **frames** [OUTPUT, Optional, default=None]

Limitations

Do NOT use: frames = decodeLabel(frames = f, ramps = r) or frames = decodeLabel(f, ramps = r) the syntax is possible, but decodeLabel will return only the last value which is ramps in this case

API details

Properties

Frames frames [INPUT, Optional, default=None]

input frames object

Ramps ramps [INPUT, Optional, default=None]

input ramps object

Frames frames [OUTPUT, Optional, default=None]
the updated frames or ramps object

History

- 1.0 18-Aug-2005 EkW Initial version of this Task
- 1.1 07-Nov-2005 JS clean up of imports
- 2.0 24-Jan-2007 MW recoded to Java
- 2.1 08-Feb-2008 MW extended to work also for Ramps

1.64. DecodeLabelTask

Full Name:	herschel.pacs.toolboxes.spg.DecodeLabelTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spg import DecodeLabelTask

Description

Decompose Label and put it into Frames Status

API Summary

Jython Syntax
`decodeLabel (Frames framesINOUT))`

Property
`Frames frames [INOUT, MANDATORY, default=NO default value]`

API details

Property

`Frames frames [INOUT, MANDATORY, default=NO default value]`
Frames class

History

- 1.0 18-Aug-2005 EkW Initial version of this Task
- 1.1 07-Nov-2005 JS clean up of imports
- 1.2 03-May-2007 EkW import for pipeline env

1.65. DecomposeDataframes

Full Name:	herschel.pacs.spg.level0.DecomposeDataframes
Type:	Java Class -
Import:	from herschel.pacs.spg.level0 import DecomposeDataframes

Description

Decompose PacsDataFrames (stored DataFrameSequence) into Frames, Ramps and store them in a PacsMix container

Extract PacsDataFrames from DataFrameSequence and convert them into Frames, Ramps. Store Frames and Ramps in PacsMix.

API Summary

Jython Syntax
`PacsMix pMi = decomposeDataframes (DataFrameSequence pacsDataFrames [,String channel = channel] [,String mode=mode] ) <p>`

Properties
`DataFrameSequence` **`pacsDataFrames`** `[INPUT, MANDATORY, default=NO default value]`
`String` **`channel`** `[INPUT, OPTIONAL, default="both"]`
`String` **`mode`** `[INPUT, OPTIONAL, default="all"]`
`Boolean` **`fullDmc`** `[INPUT, OPTIONAL, default=false]`
`PacsMix` **`pMi`** `[OUTPUT, MANDATORY, default=NO default value]`

Miscellaneous

The criterias for Frames and Ramps generation are still not fixed.

API details

Properties

`DataFrameSequence pacsDataFrames [INPUT, MANDATORY, default=NO default value]`

Container of PacsDataFrames

`String channel [INPUT, OPTIONAL, default="both"]`

channel : "red" : channel red

"blue" : channel blue

default : both

`String mode [INPUT, OPTIONAL, default="all"]`

mode : "frames" : only frames

String mode [INPUT, OPTIONAL, default="all"]
"ramps" : only ramps (subramps and rawramps) "subramps" : only subramps "rawramps" : only rawramps default : all modes

Boolean fullDmc [INPUT, OPTIONAL, default=false]
Get also the full DecMec header product

PacsMix pMi [OUTPUT, MANDATORY, default=NO default value]
PacsMix containing the generated Frames/Ramps

History

- 0.9 13-Apr-2007 EkW converted from the jython script
- 1.0 24-Jul-2007 EkW add fullDmc keyword
- 1.1 05-Oct-2007 EkW New treatment for changing mode and algorithm
- 2.0 25-Feb-2009 EkW SPR 1205

1.66. DecomposeDataframesTask

Full Name:	herschel.pacs.spg.level0.DecomposeDataframesTask
Type:	Java Task -
Import:	from herschel.pacs.spg.level0 import DecomposeDataframesTask

Description

Decompose PacsDataFrames (stored DataFrameSequence) into Frames, Ramps and store them in a PacsMix container

Extract PacsDataFrames from DataFrameSequence and convert them into Frames, Ramps. Store Frames and Ramps in PacsMix.

API Summary

Jython Syntax

```
PacsMix pMi = decomposeDataframes (DataFrameSequence
pacsDataFrames [,String channel = channel] [,String mode=mode] )
```

Properties

DataFrameSequence **pacsDataFrames** [INPUT, MANDATORY, default=NO default value]

String **channel** [INPUT, OPTIONAL, default="both"]

String **mode** [INPUT, OPTIONAL, default="all"]

Boolean **fullDmc** [INPUT, OPTIONAL, default=false]

PacsMix **pMi** [OUTPUT, MANDATORY, default=NO default value]

Miscellaneous

The criterias for Frames and Ramps generation are still not fixed.

API details

Properties

DataFrameSequence pacsDataFrames [INPUT, MANDATORY, default=NO default value]

Container of PacsDataFrames

String channel [INPUT, OPTIONAL, default="both"]

channel : "red" : channel red "blue" : channel blue default : both

String mode [INPUT, OPTIONAL, default="all"]

mode : "frames" : only frames "ramps" : only ramps (subramps and rawramps) "subramps" : only subramps "rawramps" : only rawramps default : all modes

Boolean fullDmc [INPUT, OPTIONAL, default=false]

Get also the full DecMec header product

PacsMix pMi [OUTPUT, MANDATORY, default=NO default value]
PacsMix containing the generated Frames/Ramps

History

- 0.9 13-Apr-2007 EkW converted from the jython script
- 1.0 24-Jul-2007 Ekw add fullDmc keyword

1.67. DecomposeDataframesTask

Full Name:	herschel.pacs.toolboxes.spg.DecomposeDataframesTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spg import DecomposeDataframesTask

Description

Decompose PacsDataFrames (stored DataFrameSequence) into Frames, Ramps and store them in a PacsMix container

Extract PacsDataFrames from DataFrameSequence and convert them into Frames, Ramps. Store Frames and Ramps in PacsMix.

API Summary

Jython Syntax

```
PacsMix pMi = decomposeDataframes (DataFrameSequence
pacsDataFrames [,String channel = channel] [,String mode=mode] )
```

Properties

DataFrameSequence **pacsDataFrames** [INPUT, MANDATORY, default=NO default value]
String **channel** [INPUT, OPTIONAL, default="both"]
String **mode** [INPUT, OPTIONAL, default="all"]
PacsMix **pMi** [OUTPUT, MANDATORY, default=NO default value]

Miscellaneous

The criterias for Frames and Ramps generation are still not fixed.

API details

Properties

DataFrameSequence pacsDataFrames [INPUT, MANDATORY, default=NO default value]

Container of PacsDataFrames

String channel [INPUT, OPTIONAL, default="both"]

channel : "red" : channel red "blue" : channel blue default : both

String mode [INPUT, OPTIONAL, default="all"]

mode : "frames" : only frames "ramps" : only ramps (subramps and rawramps) "subramps" : only subramps "rawramps" : only rawramps default : all modes

PacsMix pMi [OUTPUT, MANDATORY, default=NO default value]

PacsMix containing the generated Frames/Ramps

History

- 0.1 19-Apr-2005 EkW Initial version of this Task
- 0.2 04-Aug-2005 JS header adopted to jtags syntax
- 0.3 07-Sep-2005 EkW use extractFrames, extractRamps
- 0.4 07-Nov-2005 JS clean up of imports
- 0.5 05-Dec-2005 BV take out import of deprecated ccm classes
- 0.6 16-Jan-2006 RH updated according to new UNIQ interface in ia.numeric
- 0.7 23-06-2006 MW/EkW correct mode selection (midx instead of mode[0]
- 0.8 23-06-2006 EkW SPR 519
- 0.9 10-04-2007 EkW SPR 712/713

1.68. DetectCalibrationBlockTask

Full Name:	herschel.pacs.spg.DetectCalibrationBlockTask
Type:	Java Task -
Import:	from herschel.pacs.spg import DetectCalibrationBlockTask

Description

Identifies the calibration blocks and fills the CALSOURCE status entry. This task has been introduced

because only the labels are no longer a reliable source of information. Here also the chopper position and BBTYPE are taken into account.

API Summary

Properties
Frames **inFrames** [INPUT, MANDATORY, default=no default value]
Boolean **copy** [INPUT, OPTIONAL, default=false]
Frames **outFrames** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

Frames inFrames [INPUT, MANDATORY, default=no default value]

Input Frames object for which the Status table will be updated

Boolean copy [INPUT, OPTIONAL, default=false]

Copy input product if true

Frames outFrames [OUTPUT, MANDATORY, default=no default value]

Output Frames with updated Status table

1.69. Detection

Full Name:	herschel.pacs.spg.phot.deglitching.WTMML.detection.Detection
Type:	Java Class -
Import:	from herschel.pacs.spg.phot.deglitching.WTMML.detection import Detection
Category:	class

History

- 24/04/2006 : first java version

1.70. DigitsPerRampLen2DigitsPerSecTask

Full Name:	herschel.pacs.toolboxes.spec.DigitsPerRampLen2DigitsPerSecTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spec import DigitsPerRampLen2DigitsPerSecTask

Description

Convert Digits/reset interval to Digits/sec for the Spectrometer Frames

API Summary

Jython Syntax
`outFrames = digitsPerRampLen2DigitsPerSec(Frames inFrames [, int copy = <copy>])`

Properties
`Frames inFrames [INPUT, MANDATORY, default=NO default value]`
`Frames outFrames [OUTPUT, MANDATORY, default=NO default value]`
`Integer copy [INPUT, Optional, default=default value: 0]`

API details

Properties

`Frames inFrames [INPUT, MANDATORY, default=NO default value]`
Detector data are converted from digits/reset interval to digits/volts

`Frames outFrames [OUTPUT, MANDATORY, default=NO default value]`
Depending on copy keyword a copy of the Frames class with converted units or an updated Frames class

`Integer copy [INPUT, Optional, default=default value: 0]`
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

History

- 0.1 23-March-2006 RV Initial version of this Task
- 0.2 14-July-2006 RV Quantity conversion included
- 0.3 19-Jul-2006 EkW Comment out Quantity issues : Usage of Quantities is still not agreed/defined

1.71. Dmc2FineTimeTask

Full Name:	herschel.pacs.toolboxes.common.Dmc2FineTimeTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import Dmc2FineTimeTask

Description

Compute the FineTime from DecMec header entries : TMP1, TMP2 and CRDC

Combine the TMP1 and TMP2 field to microseconds to reconstruct the "start time" Add CRDC accoding to internal readout frequency of instrument. Convert the microseconds to FineTime

API Summary

Jython Syntax
`FineTime [] time = dmc2FineTime(Int1d tmp1, Int1d tmp2, Int1d crdc, String cameraId)`

Properties
`Int1d tmp1 [INPUT, MANDATORY, default=NO default value]`
`Int1d tmp2 [INPUT, MANDATORY, default=NO default value]`
`Int1d crdc [INPUT, MANDATORY, default=NO default value]`
`String cameraId [INPUT, MANDATORY, default=NO default value]`
`FineTime time [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

Int1d tmp1 [INPUT, MANDATORY, default=NO default value]
TMP1 field of DecMEc header data

Int1d tmp2 [INPUT, MANDATORY, default=NO default value]
TMP2 field of DecMEc header data

Int1d crdc [INPUT, MANDATORY, default=NO default value]
CRDC field of DecMec header data

String cameraId [INPUT, MANDATORY, default=NO default value]
String for camera identification ("PHOT" or "SPEC")

FineTime time [OUTPUT, MANDATORY, default=NO default value]
Array of FineTimes computed from DecMec header data

History

- 1.0 20-Dec-2004 EkW Initial version of this task

- 1.1 19-Jan-2005 EkW Clean up
- 1.2 03-Aug-2005 JS header adopted to jtags concept
- 1.3 07-Nov-2005 JS clean up of imports

1.72. dmc2TAI

Full Name:	herschel.pacs.toolboxes.common.dmc2TAI
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import dmc2TAI

Description

Compute the TAI (microseconds since 1958) from DecMec header entries : TMP1, TMP2 and CRDC

The commands DPUSelectTime and DPUWriteTime are selecting and setting a start time which is written to the TMP1 and TMP2 fields of the Dec/Mec headers. It is possible to construct an absolute time by adding counters (CRDC) to the start time considering an offset between setting and writing the start time. The TMP fields hold time information in Herschel CUC format, which is corrected on-board the spacecraft to TAI (epoch 1 Jan 1958). This representation uses 32-bits for seconds and 16 bits for fractional seconds. CUC times are multiples of 1/65536 sec and cannot be expressed as an exact multiple of 1 microsecond (the resolution of FineTime). Combine the TMP1 and TMP2 field to microseconds to reconstruct the "start time". Add CRDC accoding to internal readout frequency of instrument.

API Summary

Jython Syntax
`Long [] time = dmc2TAI(Int1d tmp1, Int1d tmp2, Int1d crdc, String camera)`

Properties
`Int1d tmp1 [INPUT, MANDATORY, default=NO default value]`
`Int1d tmp2 [INPUT, MANDATORY, default=NO default value]`
`Int1d crdc [INPUT, MANDATORY, default=NO default value]`
`String cameraId [INPUT, MANDATORY, default=NO default value]`
`FineTime time [OUTPUT, MANDATORY, default=NO default value]`

Miscellaneous

This offset is expected to be a number with an uncertainty depending on the system load. It might require an calibration file. Currently this offset is not considered. Between the Dec/Mec time and the packet time (see PusTmBinStruct) we have an offset. Therefore the association between HK and science data will be within an accuracy of 2 seconds. Dec/Mec time is represented in CUC format within TMP1 and TMP2 Dec/Mec entries.

API details

Properties

Int1d tmp1 [INPUT, MANDATORY, default=NO default value]
TMP1 field of DecMEc header data

Int1d tmp2 [INPUT, MANDATORY, default=NO default value]
TMP2 field of DecMEc header data

Int1d crdc [INPUT, MANDATORY, default=NO default value]
CRDC field of DecMec header data

String cameraId [INPUT, MANDATORY, default=NO default value]
String for camera identification ("PHOT" or "SPEC")

FineTime time [OUTPUT, MANDATORY, default=NO default value]
Array of Times computed from DecMec header data in Microseconds since 1958

See also

- ???

History

- 1.0 26-Apr-2004 EkW Initial version of this task
- 1.1 03-Aug-2005 JS header adopted to jtags concept

1.73. Dxset2CusTask

Full Name:	herschel.pacs.toolboxes.common.Dxset2CusTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import Dxset2CusTask

Description

Convert DetectorSelectionTable file (.dxset) to CUS procedure

API Summary

Jython Syntax
`dxset2Cus (filename, description)` `#dxset2Cus ("/home/icc/diego/2520.dxset", "your remarks")`

Properties
`String filename [INPUT, MANDATORY, default=NO default value]`
`String description [INPUT, MANDATORY, default=NO default value]`

API details

Properties

`String filename [INPUT, MANDATORY, default=NO default value]`
dxset filename

`String description [INPUT, MANDATORY, default=NO default value]`
dxset filename Examples : --------- dxset2Cus ("/home/icc/diego/2520.dxset", "Ekkis first try")

History

- 0.9 11-Apr-2005 RO First version from herschel.pacs.share.util import FileSelection
- 1.0 16-Jun-2006 EkW Convert dxsetConvert.py into this Task
- 1.1 16-Jun-2006 EkW Correction as different versions were floating around - got it by email from RO
- 1.2 18-Oct-2006 RO updated int/uint datatype

1.74. ExportPacketSequence2AsciiTask

Full Name:	herschel.pacs.toolboxes.common.ExportPacketSequence2AsciiTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import ExportPacketSequence2AsciiTask

Description

Convenience method which export all telemetry measures of a PacketSequence to Fits.

Internally HK data are grouped into HK Groups which are dumped as separate FITS files. In the default case the HK data are converted values (where possible - the rest raw) Also the Decompression and Frames/Ramps generation are internally done and the result saved in separate FITS files.

API Summary

Jython Syntax
`exportPacketSequence2Fits(seq, prefix [,rawValues=true|false} )`

Properties
`PacketSequence seq [INPUT, MANDATORY, default=NO default value]`
`String prefix [INPUT, MANDATORY, default=NO default value]`
`Boolean rawValues [INPUT, default value = false, default=no default value]`

API details

Properties

`PacketSequence seq [INPUT, MANDATORY, default=NO default value]`

The packetSequence holding the telemetry measures

`String prefix [INPUT, MANDATORY, default=NO default value]`

The prefix of the FITS export files.

`Boolean rawValues [INPUT, default value = false, default=no default value]`

Only raw values for HK data - default is converted values

See also

- ???

History

- 1.0 23-Nov-2006 from ExportPacketSequence2FFitsTask.py
- 1.1 04-Dec-2006 In case the first packet does not have FineTime method....

1.75. ExportPacketSequence2FitsTask

Full Name:	herschel.pacs.toolboxes.common.ExportPacketSequence2FitsTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import ExportPacketSequence2FitsTask

Description

Convenience method which export all telemetry measures of a PacketSequence to Fits.

Internally HK data are grouped into HK Groups which are dumped as separate FITS files. In the default case the HK data are converted values (where possible - the rest raw) Also the Decompression and Frames/Ramps generation are internally done and the result saved in separate FITS files.

API Summary

Jython Syntax
`exportPacketSequence2Fits(seq, prefix [,rawValues=true|false} [, onlyHK=traue|false] [, onlyScience=true|false])`

Properties
`PacketSequence seq [INPUT, MANDATORY, default=NO default value]`
`String prefix [INPUT, MANDATORY, default=NO default value]`
`Boolean rawValues [INPUT, default value = false, default=no default value]`
`Boolean onlyHK [INPUT, MANDATORY, default=default value = false]`
`Boolean onlyScience [INPUT, MANDATORY, default=default value = false]`

API details

Properties

`PacketSequence seq [INPUT, MANDATORY, default=NO default value]`

The packetSequence holding the telemetry measures

`String prefix [INPUT, MANDATORY, default=NO default value]`

The prefix of the FITS export files.

`Boolean rawValues [INPUT, default value = false, default=no default value]`

Only raw values for HK data - default is converted values

`Boolean onlyHK [INPUT, MANDATORY, default=default value = false]`

Export only HK data

`Boolean onlyScience [INPUT, MANDATORY, default=default value = false]`

Export only Science data

See also

- ???

History

- 2.0 01-Apr-2004 move from java version herschel.pacs.share.tools.ExportPacketRecorderArchiveToFits
- and ExportPacketRecorderarchive2Fits.py and include new data products (Frames
- , Ramps and PhotRaw)
- 2.1 25-Sep-2006 Various bug fixes
- 2.2 04-Dec-2006 In case the first packet does not have FineTime method....

1.76. ExtendBbidTask

Full Name:	herschel.pacs.spg.ExtendBbidTask
Type:	Jython Task -
Import:	from herschel.pacs.spg import ExtendBbidTask

Description

Extend the blocks flagged as scan by a user specified

number of samples before/after the block

API Summary

Jython Syntax
`frames =extendBbid(frames, bbidIn, before, after, copy=None)  `
`frames=extendbbid(frames,215131301l, 0,10)  `
`Repetitive call will keep enlarging the region unless copy is set!`

Properties
`Frames` **`frames`** `[INOUT, MANDATORY, default=Input Frames object]`
`long` **`bbidIn`** `[INOUT, MANDATORY, default=Building Block ID]`
`int` **`before`** `[INPUT, MANDATORY, default=Sample to be flagged before block]`
`int` **`after`** `[INPUT, MANDATORY, default=Sample to be flagged after block]`
`String` **`copy`** `[INPUT, OPTIONAL, default=default value : 0]`

API details

Properties

`Frames frames [INOUT, MANDATORY, default=Input Frames object]`

`long bbidIn [INOUT, MANDATORY, default=Building Block ID]`

`int before [INPUT, MANDATORY, default=Sample to be flagged before block]`

`int after [INPUT, MANDATORY, default=Sample to be flagged after block]`

`String copy [INPUT, OPTIONAL, default=default value : 0]`

Copy the input (0) and generate new output or overwrite the input (1)

History

- 1.0 20100216 EkW initial Task

1.77. Ext

Full Name:	herschel.pacs.spg.phot.deglitching.WTMML.util.Ext
Type:	Java Class -
Import:	from herschel.pacs.spg.phot.deglitching.WTMML.util import Ext
Category:	class

Description

This class gives a representation of successive maxima as a chained structure

History

- 24/04/2006 first java version

1.78. ExtList

Full Name:	herschel.pacs.spg.phot.deglitching.WTMML.util.ExtList
Type:	Java Class -
Import:	from herschel.pacs.spg.phot.deglitching.WTMML.util import ExtList

History

- 21 march 2006 : first java version

1.79. ExtractDmc

Full Name:	herschel.pacs.spg.level0.ExtractDmc
Type:	Java Class -
Import:	from herschel.pacs.spg.level0 import ExtractDmc

History

- 1.0 02-Jul-2007 EkW Initial version of this Task (converted from JYTHON

1.80. ExtractDmcTask

Full Name:	herschel.pacs.spg.level0.ExtractDmcTask
Type:	Java Task -
Import:	from herschel.pacs.spg.level0 import ExtractDmcTask

Description

Extract the full resolution DecMec Header into a Status and return a

PacsDmcProduct

History

- 1.0 02-Jul-2007 EkW Initial version of this Task

1.81. ExtractRampsTask

Full Name:	herschel.pacs.toolboxes.spec.ExtractRampsTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spec import ExtractRampsTask

Description

Extract PacsDataFrames from DataFrameSequence and convert them into a PacsMix of Ramps.

API Summary

Jython Syntax
`Ramps ramps = extractRamps (DataFrameSequence pacsDataFrames [,String channel = channel])`

Properties
`DataFrameSequence` **`pacsDataFrames`** `[INPUT, MANDATORY, default=NO default value]`
`String` **`channel`** `[INPUT, MANDATORY, default=No default value]`
`PacsMix` **`pacsmix`** `[OUTPUT, MANDATORY, default=NO default value]`

Miscellaneous

None

API details

Properties

`DataFrameSequence pacsDataFrames [INPUT, MANDATORY, default=NO default value]`

Container of PacsDataFrames

`String channel [INPUT, MANDATORY, default=No default value]`

channel : "red" : channel red "blue" : channel blue

`PacsMix pacsmix [OUTPUT, MANDATORY, default=NO default value]`

History

- 0.1 07-Sep-2005 EkW Initial version of this Task
- 0.2 05-Dec-2005 BV take out import of deprecated ccm classes
- 0.3 16-Jan-2006 RH updated according to new UNIQ interface in ia.numeric

1.82. ExtractRawDmcTask

Full Name:	herschel.pacs.toolboxes.common.ExtractRawDmcTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import ExtractRawDmcTask

Description

Extract the raw dmc

API Summary

Jython Syntax
`dmcData = extractRawDmc(seq , type)`

Properties
`PacketSequence seq [INPUT, MANDATORY, default=NO default value]`
`String type [INPUT, MANDATORY, default=NO default value]`
`In2d unknown [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`PacketSequence seq [INPUT, MANDATORY, default=NO default value]`
PacketSequence

`String type [INPUT, MANDATORY, default=NO default value]`
Type of data to extract the DMC data : PHOT_SC_BLUE, PHOT_SC_RED, SPEC_SC_BLUE, SPEC_SC_BLUE

`In2d unknown [OUTPUT, MANDATORY, default=NO default value]`
Integer2d array containing the DMC data

History

- 1.0 10-Mar-2007 EkW The first version is a prototype.

1.83. ExtractRawRampsTask

Full Name:	herschel.pacs.toolboxes.spec.ExtractRawRampsTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spec import ExtractRawRampsTask

Description

Extract raw ramps from a DataFrameSequence.

API Summary

Jython Syntax
`Ramps ramps = extractRawRamps (DataFrameSequence pacsDataFrames, String channel)`

Properties
`DataFrameSequence` **`pacsDataFrames`** `[INPUT, MANDATORY, default=NO default value]`
`String` **`channel`** `[INPUT, MANDATORY, default=No default value]`
`Ramps` **`ramps`** `[OUTPUT, MANDATORY, default=NO default value]`

Miscellaneous

None

API details

Properties

`DataFrameSequence pacsDataFrames [INPUT, MANDATORY, default=NO default value]`

Container of PacsDataFrames

`String channel [INPUT, MANDATORY, default=No default value]`

channel : "red" : channel red "blue" : channel blue

`Ramps ramps [OUTPUT, MANDATORY, default=NO default value]`

History

- 0.1 07-Sep-2005 EkW Initial version of this Task
- 0.2 12-Sep-2005 MW focus on raw ramps
- 0.3 07-Nov-2005 JS clean up of imports
- 0.4 05-Dec-2005 BV take out import of deprecated ccm classes
- 0.5 16-Jan-2006 RH updated according to new UNIQ interface in ia.numeric

1.84. ExtractSubRampsTask

Full Name:	herschel.pacs.toolboxes.spec.ExtractSubRampsTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spec import ExtractSubRampsTask

Description

Extract Subramps from a DataFrameSequence.

API Summary

Jython Syntax
`Ramps ramps = extractSubRamps (DataFrameSequence pacsDataFrames, String channel)`

Properties
`DataFrameSequence` **`pacsDataFrames`** `[INPUT, MANDATORY, default=NO default value]`
`String` **`channel`** `[INPUT, MANDATORY, default=No default value]`
`Ramps` **`ramps`** `[OUTPUT, MANDATORY, default=NO default value]`

Miscellaneous

None

API details

Properties

`DataFrameSequence pacsDataFrames [INPUT, MANDATORY, default=NO default value]`

Container of PacsDataFrames

`String channel [INPUT, MANDATORY, default=No default value]`

channel : "red" : channel red "blue" : channel blue

`Ramps ramps [OUTPUT, MANDATORY, default=NO default value]`

History

- 0.1 12-Sep-2005 MW Initial version of this Task
- 0.2 05-Dec-2005 BV take out import of deprecated ccm classes
- 0.3 16-Jan-2006 RH updated according to new UNIQ interface in ia.numeric

1.85. ExtRep

Full Name:	herschel.pacs.spg.phot.deglitching.WTMML.util.ExtRep
Type:	Java Class -
Import:	from herschel.pacs.spg.phot.deglitching.WTMML.util import ExtRep

History

- 13 march 2006 : first java version
- 07/03/2007 : interpolate maxima line coordinate in computeExtList

1.86. FilterOnTargetTask

Full Name:	herschel.pacs.spg.phot.FilterOnTargetTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import FilterOnTargetTask
Category:	Task/PACS Task

Description

unknown

API Summary

Jython Syntax
`frames = filterOnTarget(frames)`

Property
`Frames; MANDATORY frames [INPUT, null, default=no default value]`

API details

Property

`Frames; MANDATORY frames [INPUT, null, default=no default value]`

1.87. FilterSlewTask

Full Name:	herschel.pacs.spg.phot.FilterSlewTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import FilterSlewTask
Category:	Task/PACS Task

Description

unknown

API Summary

Jython Syntax
`framesOut = filterSlew(frames)`

Properties
`Frames; MANDATORY frames [INPUT, null, default=no default value]`
`a filtered copy of framesIn ! framesOut [OUTPUT, MANDATORY, default=no default value]`

API details

Properties

`Frames; MANDATORY frames [INPUT, null, default=no default value]`

`a filtered copy of framesIn ! framesOut [OUTPUT, MANDATORY, default=no default value]`

1.88. FindBlocksTask

Full Name:	herschel.pacs.spg.FindBlocksTask
Type:	Java Task -
Import:	from herschel.pacs.spg import FindBlocksTask

Description

Finds the logical blocks in which the observation is organized. The selection of blocks is based

on the logic of the AOT and is based mainly on the observational building block type (BBTYPE) and the labels used in the decmec sequences. Each block gets a human readable ID which tells whether it is e.g. a Calibration Source measurement, a up or down scan, etc.

API Summary

Jython Syntax
`Frames framesOut = findBlocks (frames)`

Properties
`Frames` **`frames`** `[INPUT, MANDATORY, default=NO default value]`
`CalibrationTree` **`calTree`** `[INPUT, OPTIONAL, default=NO default value]`
`ObcpDescription` **`obcpDesc`** `[INPUT, OPTIONAL, default=NO default value]`
`Frames` **`framesOut`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Frames frames [INPUT, MANDATORY, default=NO default value]`
Input Frames product

`CalibrationTree calTree [INPUT, OPTIONAL, default=NO default value]`
Calibration Tree

`ObcpDescription obcpDesc [INPUT, OPTIONAL, default=NO default value]`
OBCP description calibration product, which contains the table of block types

`Frames framesOut [OUTPUT, MANDATORY, default=NO default value]`
Frames with ObcpBlockTable holding the block information

History

- `2009-08-14 - jdejong`: 1.24 SPR 1934: added exception for OCBP 27 in nodpos evaluation

- 2009-07-20 - jdejong: 1.23 SCR 1753: added OBCP 35 calibration block identification
- 2009-07-16 - jdejong: 1.22 SPR 1803: new task detectCalibrationBlock
- 2009-07-16 - jdejong: 1.21 Implemented SCR 1753: OBCP -> BBID for block recognition
- 2009-06-24 - jdejong: 1.20 SPR 1737, SPR 1739
- 2009-05-25 - jdejong: 1.19 SPR 1637: filling new blocktable columns RasterLineNum and RasterColumnNum
- 2009-04-28 - jdejong: 1.18 SPR 1434: switched the nodpos on/off source label assumption in findBlocks
- 2009-04-22 - jdejong: 1.17 SPR 1533
- 2009-03-25 - jdejong: 1.16 SPR 1462
- 2009-03-05 - jdejong: 1.15 SCR 1384, SCR 1385
- 2009-03-04 - jdejong: 1.14 SCR 1378, SCR 1381
- 2009-03-03 - jdejong: 1.13 SPR 1097: implemented on/off source swapping with nod position also for photometer frames
- 2009-02-26 - jdejong: 1.12 SPR 1371: added grating position based line identification
- 2009-02-24 - jdejong: 1.11 limited label setting to spectrometer blocks
- 2009-02-24 - jdejong: 1.10 SPR 1097: implemented nodding column filling and nod dependent label handling
- 2008-07-31 - jdejong: 1.9 introduced allow NULLs for two parameters
- 2008-07-17 - jdejong: 1.8 Added extra parameter to provide calibration product directly
- 2008-07-16 - rik: 1.7 D_PACS_SPG_200_18
- 2008-06-26 - jdejong: 1.6 Added label onsource offsource label columns to the block table
- 2008-06-25 - jdejong: 1.5 Added Block IDs which are stored in an enumeration
- 2008-06-23 - jdejong: 1.4 Equal treatment of OBCPs 0 and 63 and code cleanup
- 2008-06-10 - jdejong: 1.3 Made calTree parameter optional
- 2008-06-09 - jdejong: 1.2 Introduced calibration product for the block description table (part of the ObcpDescription product).
- 2008-05-21 - jdejong: 1.1 Translated from Jython to Java and refactored
- 2008-02-22 - michael: 1.18 changed IO statements to IN and added OUT to FindBlocksTask
- 2007-04-30 - ewieprec: 1.17 as the bug in UNIQ was corrected, the bug fix did not work any more...
- 2006-09-04 - jschreiber: 1.16 fix of task framework change: INOUT to IO in spg
- 2006-08-31 - ewieprec: 1.15 correct java imports
- 2006-08-30 - jschreiber: 1.14 Tasks: mode = ... replaced by type = IN/OUT/IO
- 2006-08-04 - ewieprec: 1.13 add OBCP 63 - no OBCP active

- 2006-07-27 - ewieprec: 1.12 Use Float.intValue() for obcp assignment
- 2006-07-20 - ewieprec: 1.11 improved for Spectrometer - still no testdata available !
- 2006-07-11 - ewieprec: 1.10 first version
- 2006-07-06 - ewieprec: 1.9 first version following Thomas new instructions
- 2006-01-16 - rik: 1.8 updated according to new UNIQ interface in ia.numeric
- 2005-12-06 - bart: 1.7 Removed import statement of deprecated ccm class
- 2005-11-07 - jschreiber: 1.6 clean up of imports according to AI of ICC#22
- 2005-10-21 - ewieprec: 1.5 for THomas
- 2005-10-17 - ewieprec: 1.4 correct for Spectrometer
- 2005-08-19 - ewieprec: 1.3 Adopt to new binary struct definition
- 2005-08-18 - ewieprec: 1.2 working prototype
- 2005-08-12 - ewieprec: 1.1 prototype to evaluate with Thomas

1.89. FindFilesTask

Full Name:	herschel.pacs.toolboxes.common.FindFilesTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import FindFilesTask

Description

Find filenames considering a certain pattern. walk through su-directories from start directory (may be switched off)

API Summary

Jython Syntax
`String1d filenames = findFiles(String startDirectory, String pattern [,Boolean noWalk])`

Properties
`String startDirectory [INPUT, MANDATORY, default=No default value]`
`Int1d pattern [INPUT, OPTIONAL, default=Default value *.*]`
`Boolean noWalk [INPUT, OPTIONAL, default=Default value False]`
`String1d filenames [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`String startDirectory [INPUT, MANDATORY, default=No default value]`

Start Directory

`Int1d pattern [INPUT, OPTIONAL, default=Default value *.*]`

Filename pattern

`Boolean noWalk [INPUT, OPTIONAL, default=Default value False]`

Walk through sub directories

`String1d filenames [OUTPUT, MANDATORY, default=NO default value]`

returned filenames with directories

History

- 1.0 23-Jan-2007 EkW Initial version of this task

1.90. fitRampPolynomial

Full Name:	herschel.pacs.toolboxes.numeric.fitRampPolynomial
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import fitRampPolynomial

Description

Jython function to fit a user defined polynomial to given ramp

This function fits a polynomial to the given x, y data of a Pacs ramp, you can reject a defined number of first and last values of the ramp

API Summary

Jython Syntax
`TableDataset parameter = fitRampPolynomial(Double1d xpoint, Double1d ypoint, int poly_degree, int n_firstReject, int n_lastReject)`

Properties
`Double1d` **`xpoint`** `[INPUT, MANDATORY, default=NO default value]`
`Double1d` **`ypoint`** `[INPUT, MANDATORY, default=NO default value]`
`int` **`poly_degree`** `[INPUT, MANDATORY, default=NO default value]`
`int` **`n_firstReject`** `[INPUT, MANDATORY, default=NO default value]`
`int` **`n_lastReject`** `[INPUT, MANDATORY, default=NO default value]`
`TableDataset` **`parameter`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Double1d xpoint [INPUT, MANDATORY, default=NO default value]`
x-y arrays of the ramp values, length of x and y arrays must be equal!

`Double1d ypoint [INPUT, MANDATORY, default=NO default value]`
x-y arrays of the ramp values, length of x and y arrays must be equal!

`int poly_degree [INPUT, MANDATORY, default=NO default value]`
polynomial degree of the fit polynomial

`int n_firstReject [INPUT, MANDATORY, default=NO default value]`
number of discarded first values of the ramp

`int n_lastReject [INPUT, MANDATORY, default=NO default value]`
number of discarded last values of the ramp

TableDataset parameter [OUTPUT, MANDATORY, default=NO default value]
first Column: fitted polynomial parameters, number of parameters is degree + 1 second Column: errors of the fitted polynomial parameters

History

- 1.0 20050511 JS initial version
- 1.1 20050524 JS imports moved inside function
- 1.2 20050629 JS have to import toolbox.fit for PolynomialModel
- 1.3 20050803 JS header adopted to jtags concept
- 1.4 20050829 JS return after error deleted
- 1.5 20051031 JS bypass implemented for known bug fitter.getStandardDeviation (SPR 1311)
- 1.6 20051115 JS another bypass for SPR 1311 implemented according to proposal of BV
- 1.7 20060130 JS return None if input parameters are wrong

1.91. FitRampsTask

Full Name:	herschel.pacs.spg.spec.FitRampsTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import FitRampsTask

Description

function that converts a Ramps object containing ramps or averaged

ramps to a Frames object after fitting the ramps linearly. This function fits linearly the given ramps and returns the according Frames object containing the slopes as signals. Masks are applied as follows: a master mask is constructed using the already available active masks in the Ramps object. The master mask is then applied to the indices and readouts before the signal is calculated. As x-axis the fine time information per sample is used. This works also for averaged ramps (sub-ramps). The units of the output slopes is corresponding to the onboard fitting readouts per reset interval. The status parameters are also populated using the first elements of the ramp status arrays or the mean where the masked resets are discarded before. The stdev array is populated using the uncertainty of the linear fit coefficient.

API Summary

Jython Syntax
`Frames outFrame = fitRamps(Ramps inRamp [, int degree = <degree>] [, int firstReject = <firstReject>] [, int lastReject = <lastReject>])`

Properties
`Ramps inRamp [INPUT, MANDATORY, default=NO default value]`
`int degree [INPUT, OPTIONAL, default=1]`
`int firstReject [INPUT, OPTIONAL, default=0]`
`int lastReject [INPUT, OPTIONAL, default=0]`

API details

Properties

Ramps inRamp [INPUT, MANDATORY, default=NO default value]
input Ramps object

int degree [INPUT, OPTIONAL, default=1]
optional: degree of fitted polynomial, default: 1

int firstReject [INPUT, OPTIONAL, default=0]
optional: number of readouts at the start of the ramp that are not considered for fitting, default: 0

int lastReject [INPUT, OPTIONAL, default=0]
optional: number of readouts at the end of the ramp that are not considered for fitting, default: 0

History

- 1.0 20050511 JS initial version
- 1.1 20050513 JS get the x-axis from fine time to be able to
- consider averaged sub-ramps
- 1.2 20050524 JS fine time replaced by crcrmp values, imports shifted inside function
- 1.3 20050620 JS mask handling introduced, crcrmp values replaced by indices
- 1.4 20050804 JS header adopted to jtags syntax
- 1.5 20050915 JS import of pacs.signal was missing
- 1.6 20051213 JS major improvements, correct handling of time and empty ramps, populating status parameters
- 1.7 20051214 JS conversion to task
- 1.8 20060303 JS Ramps.getRampValues to getSignal()
- 1.9 20060316 JS Stdev of fit is filled in Frames and memory handling improved
- camera parameter is read directly from Ramps, not input parameter anymore
- 1.10 20060327 JS status parameter conversion distinguishes 1d and 2d arrays
- 1.11 20060518 JS bug fix
- 1.12 20060720 EkW converted to java
- 1.13 20070111 JS jtags header added
- 1.14 20080519 JS implementation of SPR 817, additional optional parameter
- 1.15 20080611 JS implementation of SPR 887, optional boolean parameters ignore...Mask added
- 1.16 20081010 JS implementtion of SPR 1080

1.92. FlagChopMoveFramesTask

Full Name:	herschel.pacs.spg.FlagChopMoveFramesTask
Type:	Java Task -
Import:	from herschel.pacs.spg import FlagChopMoveFramesTask

Description

SAME as CleanPlateauFramesTask! -> see CleanPlateauFramesTask URM entry!

This task flags the unreliable signals at the beginning of a chopper plateau.

(A plateau is the chopper on or off or calibration source position inclusive the transition!) Therefore it calculates the differences of the median of a plateau to the single readouts and compares with the maximum tolerance read from the cal file chopJitterThreshold.

1.93. FlagChopMoveRampsTask

Full Name:	herschel.pacs.toolboxes.spg.FlagChopMoveRampsTask
Type:	Java Task -
Import:	from herschel.pacs.toolboxes.spg import FlagChopMoveRampsTask

Description

SAME as CleanPlateauRampsTask! -> see CleanPlateauRampsTask URM entry!

This task flags the unreliable signals at the beginning of a chopper plateau.

(A plateau is the chopper on or off or calibration source position inclusive the transition!) Therefore it calculates the differences of the median of a plateau to the single readouts and compares with the maximum tolerance read from the cal file chopJitterThreshold.

1.94. FlagDeviatingOpenDummyRampsTask

Full Name:	herschel.pacs.spg.spec.FlagDeviatingOpenDummyRampsTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.spec import FlagDeviatingOpenDummyRampsTask

Description

Jython spectrometer related task that observes the behaviour of open and dummy channel ramps

and sets the valid flag to non-valid for the corresponding module if the ramps show weird/deviating behaviour. Such a behaviour like non-zero or oscillating ramps in the open channel would indicate e.g. saturation or other problems in the FEE electronics. It might be necessary to introduce a calibration file that contains the normal noise thresholds of the ramps of the dummy and the open channels to be able to detect non-normal deviations. A closer investigation of the open/dummy signals revealed cross-talk effects on the about 3% level, e.g. chopped and non-chopped measurements reveal different standard deviations (factor about 100!), e.g. the chopper cycle is visible in the signal course of the open/dummy channel! Also major differences in signal level introduced by different power supply groups are established. This means we cannot use simply a long term noise values for each channel stored in a cal file but we have to determine a global noise level for each measurement distinguished by power supply groups. This presumes that most of the modules and/or resets show normal behaviour.

API Summary

Jython Syntax
`Ramps outRamp = flagDeviatingOpenDummyRamps(Ramps inRamp [, Double sigma][, int copy = <copy>])`

Properties
`Ramps inRamp [INPUT, MANDATORY, default=NO default value]`
`Integer sigma [INPUT, Optional, default=default value: 3.0]`
`Integer copy [INPUT, Optional, default=default value: 0]`
`Ramps outRamp [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

Ramps inRamp [INPUT, MANDATORY, default=NO default value]
input Ramps object

Integer sigma [INPUT, Optional, default=default value: 3.0]
factor to standard deviaton within the signals are valid

Integer copy [INPUT, Optional, default=default value: 0]
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

Ramps outRamp [OUTPUT, MANDATORY, default=NO default value]
returned Ramps

History

- 1.0 20060531 JS initial template
- 1.1 20070829 JS prototype
- 1.2 20080221 JS input parameter calVersion deleted
- 1.3 20080527 JS improved version distinguishes power supply groups

1.95. FlagDeviatingOpenDummySignalsTask

Full Name:	herschel.pacs.spg.spec.FlagDeviatingOpenDummySignalsTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.spec import FlagDeviatingOpenDummySignalsTask

Description

Jython spectrometer related task that observes the behaviour of open and dummy channel signals

and sets the valid flag to non-valid for the corresponding module if the signals show weird/deviating behaviour. Such a behaviour like non-zero or spiking signals in the open channel would indicate e.g. saturation or other problems in the FEE electronics. It might be necessary to introduce a calibration file that contains the normal noise thresholds of the signals of the dummy and the open channels to be able to detect non-normal deviations. A closer investigation of the open/dummy signals revealed cross-talk effects on the about 3% level, e.g. chopped and non-chopped measurements reveal different standard deviations (factor about 100!), e.g. the chopper cycle is visible in the signal course of the open/dummy channel! Also major differences in signal level introduced by different power supply groups are established. This means we cannot use simply a long term noise values for each channel stored in a cal file but we have to determine a global noise level for each measurement distinguished by power supply groups. This presumes that most of the modules and/or resets show normal behaviour.

API Summary

Jython Syntax
`Frames outFrame = flagDeviatingOpenDummySignals(Frames inFrame, [Double sigma = 3.][, int copy = <copy>])`

Properties
`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
`Double sigma [INPUT, Optional, default=default value: 3.]`
`Integer copy [INPUT, Optional, default=default value: 0]`
`Frames outFrame [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

Frames inFrame [INPUT, MANDATORY, default=NO default value]
input Frames object

Double sigma [INPUT, Optional, default=default value: 3.]
factor to sigma within which signals deviations are valid

Integer copy [INPUT, Optional, default=default value: 0]
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

Frames outFrame [OUTPUT, MANDATORY, default=NO default value]
returned Frames

History

- 1.0 20060531 JS initial template
- 1.1 20070829 JS prototype
- 1.2 20080221 JS input parameter calVersion deleted
- 1.3 20080527 JS improved version distinguishes power supply groups

1.96. FlagGratMoveFramesTask

Full Name:	herschel.pacs.spg.spec.FlagGratMoveFramesTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import FlagGratMoveFramesTask

Description

This task flags signals taken while the grating was moving.

These unreliable signals are found by comparing the grating steps with a threshold stored in the cal file GratingJitterThreshold.

API Summary

Jython Syntax
`Frames outFrame = flagGratMoveFrames(Frames inFrame, PacsDmcProduct dmcHead[,PacsCal calTree]` `[, GratingJitterThreshold thresh][,QualityContext qualContext] [, int copy])`

Properties
`Frames` **`inFrame`** `[INPUT, Mandatory, default=no default value]`
`PacsDmcProduct` **`dmcHead`** `[INPUT, Mandatory, default=no default value]`
`PacsCal` **`calTree`** `[INPUT, Optional, default=null]`
`GratingJitterThreshold` **`gratingJitterThreshold`** `[INPUT, Optional, default=null]`
`Integer` **`algNo`** `[INPUT, Optional, default=1]`
`Integer` **`copy`** `[INPUT, Optional, default=0]`
`QualityContext` **`qualityContext`** `[INOUT, Optional, default=null]`
`Frames` **`outFrame`** `[OUTPUT, Mandatory, default=no default value]`

API details

Properties

`Frames inFrame [INPUT, Mandatory, default=no default value]`
input Frames object

`PacsDmcProduct dmcHead [INPUT, Mandatory, default=no default value]`
full resolution decmec header

`PacsCal calTree [INPUT, Optional, default=null]`
calfile access

GratingJitterThreshold gratingJitterThreshold [INPUT, Optional, default=null]

grating position jitter threshold cal file

Integer algNo [INPUT, Optional, default=1]

algNo 1 = Juergen's simple (default), 2 = A probable mis-representation of DF's alg.

Integer copy [INPUT, Optional, default=0]

indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

QualityContext qualityContext [INOUT, Optional, default=null]

quality control description

Frames outFrame [OUTPUT, Mandatory, default=no default value]

Frames object containing the mask "GRATMOVE" for frames on grating transitions

1.97. FlagGratMoveRampsTask

Full Name:	herschel.pacs.spg.spec.FlagGratMoveRampsTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import FlagGratMoveRampsTask

Description

This task flags signals taken while the grating was moving.

These unreliable signals are found by comparing the grating steps with a threshold stored in the cal file GratingJitterThreshold.

API Summary

Jython Syntax
`Ramps outRamp = flagGratMoveRamps(Ramps inRamp, PacsDmcProduct dmcHead[,PacsCal calTree]`
`[, GratingJitterThreshold thresh][,QualityContext qualContext] [, int copy])`

Properties
`Ramps` **`inRamp`** `[INPUT, Mandatory, default=no default value]`
`PacsDmcProduct` **`dmcHead`** `[INPUT, Mandatory, default=no default value]`
`PacsCal` **`calTree`** `[INPUT, Optional, default=null]`
`GratingJitterThreshold` **`gratingJitterThreshold`** `[INPUT, Optional, default=null]`
`Integer` **`algNo`** `[INPUT, Optional, default=1]`
`Integer` **`copy`** `[INPUT, Optional, default=0]`
`QualityContext` **`qualityContext`** `[INOUT, Optional, default=0]`
`Ramps` **`outRamp`** `[OUTPUT, Mandatory, default=no default value]`

API details

Properties

`Ramps inRamp [INPUT, Mandatory, default=no default value]`
input Ramps object

`PacsDmcProduct dmcHead [INPUT, Mandatory, default=no default value]`
full resolution decmec header

`PacsCal calTree [INPUT, Optional, default=null]`
calfile access

GratingJitterThreshold gratingJitterThreshold [INPUT, Optional, default=null]

grating position jitter threshold cal file

Integer algNo [INPUT, Optional, default=1]

algNo 1 = Juergen's simple (default), 2 = A probable mis-representation of DF's alg.

Integer copy [INPUT, Optional, default=0]

indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

QualityContext qualityContext [INOUT, Optional, default=0]

quality control description

Ramps outRamp [OUTPUT, Mandatory, default=no default value]

Ramps object containing the mask "GRATMOVE" for frames on grating transitions

1.98. fpuAngle2ChopPosPol

Full Name:	herschel.pacs.toolboxes.common.fpuAngle2ChopPosPol
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import fpuAngle2ChopPosPol

Description

Convert chopper angle wrt FPU to chopper digital readback units using polynomial fit parameters

API Summary

Jython Syntax
`Double chopperPosition = fpuAngle2ChopPosPol(Double1d fpuAngle [, redundant = 0][,CalibrationTree calTree][,ChopperAngle chopperAngle][,ChopperAngleRedundant chopperAngleRedundant] [,String model][,FineTime time])`

Properties
`Double1d` **`fpuAngle`** `[INPUT, MANDATORY, default=NO default value]`
`int` **`redundant`** `[INPUT, Optional, default=default value: 0]`
`CalibrationTree` **`calTree`** `[INPUT, Optional, default=default value: None]`
`ChopperAngle` **`chopperAngle`** `[INPUT, Optional, default=default value: None]`
`ChopperAngleRedundant` **`chopperAngleRedundant`** `[INPUT, Optional, default=default value: None]`
`String` **`model`** `[INPUT, Optional, default=default value: "FM"]`
`FineTime` **`time`** `[INPUT, Optional, default=default value: present date]`
`Double1d` **`chopperPosition`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Double1d fpuAngle [INPUT, MANDATORY, default=NO default value]`
Chopper physical angle (i.e. wrt FPU, not on the sky) in degrees

`int redundant [INPUT, Optional, default=default value: 0]`
if 0: nominal field plate calibration is used (FPI), if 1: redundant one is used (FPII)

`CalibrationTree calTree [INPUT, Optional, default=default value: None]`
cal file access

ChopperAngle chopperAngle [INPUT, Optional, default=default value: None]
chopper calibration readouts to fpu angle cal file access

ChopperAngleRedundant chopperAngleRedundant [INPUT, Optional, default=default value: None]
chopper calibration for redundant FP readouts to fpu angle cal file access

String model [INPUT, Optional, default=default value: "FM"]
model name FM, FS or QM

FineTime time [INPUT, Optional, default=default value: present date]
time of measurement

Double1d chopperPosition [OUTPUT, MANDATORY, default=NO default value]
Chopper readback position

History

- 08/05/2006 JS Created
- 11/05/2006 JS imports updated
- 13/06/2006 JS reworked to be able to use 1D arrays
- 26/06/2006 JS strange behaviour of hcss adopted
- lambda x of where suddenly only accepts global vars?
- import * not accepted inside functions anymore
- 05/07/2006 JS imports placed inside functions despite warning
- 11/09/2006 JS where function updated to get rid of warnings
- 28/11/2006 JS option for redundant FPII calibration introduced
- 22/11/2007 JS adapt to new cal framework
- 22/02/2008 JS adapt again to new cal framework

1.99. framesL05

Full Name:	herschel.pacs.spg.pipeline.ipipe.phot.framesL05
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.ipipe.phot import framesL05

History

- 1.0 20090313 EkW
- 2.0 20100119 EkW Prepare for Operations

1.100. framesL05

Full Name:	herschel.pacs.spg.pipeline.ipipe.spec.framesL05
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.ipipe.spec import framesL05

History

- 1.0 20090313 JS initial version
- 1.1 20090624 JS updated
- 1.2 20090716 JS introduce slicing

1.101. Gauss2DFitTask

Full Name:	herschel.pacs.toolboxes.numeric.Gauss2DFitTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import Gauss2DFitTask

Description

Jython task that will do a two-dimensional gaussfit to an image aource

API Summary

Jython Syntax
`result = dl.gaussfit(array,guess,radius=3,invert=None,mask=None)`
`array | in | Double2d input image`
`guess | in | Double1d(4) approximate peak,x,y,sigma as initial guess`
`radius | in | int 'radius' of centroid box in pixel, default 4`
`invert | in | any if set, invert before centroiding (negative beams!)`
`mask | in | Double2d if set, mask array - 0 for bad, 1 for good pixels, same dimensions as input array`
`result | return | Double1d(4) resulting peak, x,y and sigma of two-d gaussian`

Properties
Double2d **array** [INPUT, MANDATORY, default=NO default Value]
Double1d **guess** [INPUT, MANDATORY, default=NO default value]
Int **radius** [INPUT, OPTIONAL, default=default value: 3]
String **invert** [INPUT, OPTIONAL, default=default value: None]
Double2d **mask** [INPUT, OPTIONAL, default=default value: None]
Double1d **result** [OUTPUT, MANDATORY, default=No default value]

API details

Properties

Double2d array [INPUT, MANDATORY, default=NO default Value]
input image

Double1d guess [INPUT, MANDATORY, default=NO default value]
approximate guess for peak, x, y, sigma

Int radius [INPUT, OPTIONAL, default=default value: 3]
'radius' of centroid box in pixel

String invert [INPUT, OPTIONAL, default=default value: None]
if set, invert before centroiding (negative beams!)

Double2d mask [INPUT, OPTIONAL, default=default value: None]
if set, mask array - 0 for bad, 1 for good

Double1d result [OUTPUT, MANDATORY, default=No default value]
Resulting peak, x, y and sigma value of best-fit Gaussian

History

- 1.0 20040928 DL first version
- 1.1 20041215 DL add mask option, better use of jython keywords
- 1.2 20050314 DL adapted numeric import
- 1.3 18-Aug-2006 BA converted to Task
- 1.4 31-Aug-2006 EkW correct java imports

1.102. getCalPool

Full Name:	herschel.pacs.cal.util.GetCalPool
Alias:	getCalPool
Type:	Jython Task -
Import:	from herschel.pacs.cal.util import getCalPool
Category:	PACS/Calibration

Description

Get the default Calibration Pool.

This function uses the default settings for poolName and poolLocation if these arguments are not given. Defaults are taken from the properties `hcss.pacs.cal.store.id` and `hcss.pacs.cal.store.dir`.

Example

Example 1: inspect your private calibration storage

```
result = browseProduct(ProductStorage(getCalPool(poolName="myCalFlightSpare",
poolLocation="/home/rik/pools")))
```

API Summary

Jython Syntax

```
calpool = getCalPool([poolName=<id>][, poolLocation=<directory>][,
verbose=False,True])
```

Properties

`String` **`poolName`** `[INPUT, OPTIONAL, default=loaded from hcss.pacs.cal.store.id]`
`String` **`poolLocation`** `[INPUT, OPTIONAL, default=loaded from hcss.pacs.cal.store.dir]`
`Boolean` **`verbose`** `[INPUT, OPTIONAL, default=False]`

API details

Properties

`String poolName [INPUT, OPTIONAL, default=loaded from hcss.pacs.cal.store.id]`

Specify an alternative name for the calibration pool

`String poolLocation [INPUT, OPTIONAL, default=loaded from hcss.pacs.cal.store.dir]`

Specify an alternative name for the calibration pool

`Boolean verbose [INPUT, OPTIONAL, default=False]`

Provide more information on the location and name of the calibration pool

See also

- getCalPool

1.103. getCalProduct

Full Name:	herschel.pacs.cal.util.GetCalProduct
Alias:	getCalProduct
Type:	Jython Task -
Import:	from herschel.pacs.cal.util import getCalProduct
Category:	PACS/Calibration

Description

Get a specific version for a calibration product.

Use this method when you want to load a specific version of a calibration product regardless of the version in the calibration tree.

The requested version is loaded from the calibration storage that is returned by the getCalStorage() function. Refer to the documentation of getCalStorage in order to know from which Pool that calibration product is loaded exactly.

Example

Example 1: load the first version of the ArrayInstrument calibration product

```
ai = getCalProduct("Spectrometer", -"ArrayInstrument", 1)
ca = getCalProduct("Common", -"ChopperAngle", 2, -"QM")
```

API Summary

Jython Syntax

```
cal = getCalProduct(unit, name, version [, model="FM"])
```

Properties

- `String unit [INPUT, MANDATORY, default=no default]`
- `String name [INPUT, MANDATORY, default=no default]`
- `Integer version [INPUT, MANDATORY, default=no default]`
- `String model [INPUT, OPTIONAL, default=FM]`

API details

Properties

`String unit [INPUT, MANDATORY, default=no default]`

Accepted values are Common, Spectrometer, Photometer (Capitalized)

`String name [INPUT, MANDATORY, default=no default]`

The class name for the calibration Product (CamelCase)

`Integer version [INPUT, MANDATORY, default=no default]`

An integer representing the version of the cal product

String model [INPUT, OPTIONAL, default=FM]
Accepted values are FM, FS, QM (optional argument)

See also

- getCalStorage

1.104. getCalStorage

Full Name:	herschel.pacs.cal.util.GetCalStorage
Alias:	getCalStorage
Type:	Jython Task -
Import:	from herschel.pacs.cal.util import getCalStorage
Category:	PACS/Calibration

Description

Get the default calibration pool as a product storage.

The default calibration storage is a wrapper around the default calibration pool. See getCalPool() for details on the exact location and name of the calibration pool used as a default.

Example

Example 1: inspect the default calibration storage

```
result = browseProduct(getCalStorage())
```

See also

- getCalPool

1.105. getCalTree

Full Name:	herschel.pacs.cal.GetPacsCalTask
Alias:	getCalTree
Type:	Java Task -
Import:	from herschel.pacs.cal.all import getCalTree
Category:	PACS/Calibration Framework

Description

Task to retrieve the PACS calibration tree.

This task allows easy access to the calibration tree. All parameters are optional, when called without any parameters it loads and initializes the calibration tree for the defaults as defined by the PacsConfig class, i.e. the Flight Model (FM) and scope BASE.

The different models that are recognized are: FM (Flight Model), QM (Qualification Model), and FS (Flight Spare).

Example

Example 1: a few examples of optional and positional parameters

```
fm = getCalTree()
calTree = getCalTree("FM", -"BASE")
calTree = getCalTree(model="QM", scope="TEST", time="17-Oct-2004 12:09:00")
calTree = getCalTree(model="QM", scope="TEST", obs=myObservation)
```

API Summary

Jython Syntax

```
calTree = getCalTree(model="FM", scope="BASE", time=time, obs=obs)
```

Properties

String **model** [INPUT, OPTIONAL, default=DEFAULT_MODEL]
String **scope** [INPUT, OPTIONAL, default=DEFAULT_SCOPE]
String **time** [INPUT, OPTIONAL, default=null]
Boolean **verbose** [INPUT, OPTIONAL, default=false]
ObservationContext **obs** [INPUT, OPTIONAL, default=null]

API details

Properties

String model [INPUT, OPTIONAL, default=DEFAULT_MODEL]

The instrument model as a string, the following models are recognized: "FM", "QM", "FS".

String scope [INPUT, OPTIONAL, default=DEFAULT_SCOPE]

The scope for which this calibration tree is valid, i.e. BASE is considered stable and fully tested, while TEST is used by the ICCs and not yet validated.

String time [INPUT, OPTIONAL, default=null]

The time for which the "time dependent" calibration products in the tree will be selected. The time must be specified in the format: "dd-MMM-yyyy HH:mm:ss".

Boolean verbose [INPUT, OPTIONAL, default=false]

Be verbose, i.e. print information about the location of the calibration store on the console.

ObservationContext obs [INPUT, OPTIONAL, default=null]

The observation context for which the calibration tree should be selected. The start date of the observation is picked up in order to select the correct "time dependent" calibration products.

1.106. GetDataBlocksIdx

Full Name:	herschel.pacs.spg.GetDataBlocksIdx
Type:	Java Class -
Import:	from herschel.pacs.spg import GetDataBlocksIdx

Description

Extract start and end of true blocks in an Bool1d

API Summary

Jython Syntax
`TableDataset indexTable = getDataBlocksIdx(Bool1d array)`

Property
`Bool1d array [INPUT, MANDATORY, default=NO default value]`

API details

Property

`Bool1d array [INPUT, MANDATORY, default=NO default value]`
Boolean array from where "true blocks" are identified

1.107. GetDataBlocksIdxTask

Full Name:	herschel.pacs.spg.GetDataBlocksIdxTask
Type:	Java Task -
Import:	from herschel.pacs.spg import GetDataBlocksIdxTask

Description

Extract start and end of true blocks in an Bool1d

in getDataBlocksIdx

API Summary

Jython Syntax
`TableDataset indexTable = getDataBlocksIdx(Bool1d array)`

Property
`Bool1d array [INPUT, MANDATORY, default=NO default value]`

API details

Property

`Bool1d array [INPUT, MANDATORY, default=NO default value]`
Boolean array from where "true blocks" are identified

1.108. getDutyCycle

Full Name:	herschel.pacs.toolboxes.common.getDutyCycle
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import getDutyCycle

Description

Jython function that calculates the flat part (flat plateaux) of a input chopper cycle

(defined by 2 zero transitions) using the plateau accuracy specification thresholds.

It also determines the standard deviation of the flat plateau readouts and the chopper frequency (but only of a half cycle!).

API Summary

Jython Syntax
`JythonList duty = getDutyCycle(Double1d chopcycle, Double setpoint, Double threshold, Double readoutFreq)`

Properties
`Double1d chopcycle [INPUT, MANDATORY, default=NO default value]`
`Double setpoint [INPUT, MANDATORY, default=NO default value]`
`Double threshold [INPUT, MANDATORY, default=NO default value]`
`Double readoutFreq [INPUT, MANDATORY, default=NO default value]`
`Double duty[0] [OUTPUT, MANDATORY, default=NO default value]`
`Double duty[1] [OUTPUT, MANDATORY, default=NO default value]`
`Double duty[2] [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Double1d chopcycle [INPUT, MANDATORY, default=NO default value]`

array of one (averaged) chopper cycle (between 2 zero transitions)

`Double setpoint [INPUT, MANDATORY, default=NO default value]`

input commanded chopper deflection

`Double threshold [INPUT, MANDATORY, default=NO default value]`

plus/minus threshold of plateau accuracy, max deviation from commanded setpoint

`Double readoutFreq [INPUT, MANDATORY, default=NO default value]`

readout frequency of chopper readouts (different for spectrometer/photometer)

`Double duty[0] [OUTPUT, MANDATORY, default=NO default value]`

duty cycle in percent of modulation function

Double duty[1] [OUTPUT, MANDATORY, default=NO default value]
standard deviation of readouts of flat plateau part

Double duty[2] [OUTPUT, MANDATORY, default=NO default value]
chopper frequency of cycle between 2 zero transitions (not the real chopper modulation frequency!)

See also

- ???

History

- 1.0 20060420 JS initial version
- 1.1 20060511 JS additional input parameter readout frequency
- 1.2 20060517 JS import added at beginning

1.109. GetEffectiveCapacitanceTask

Full Name:	herschel.pacs.toolboxes.spec.GetEffectiveCapacitanceTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spec import GetEffectiveCapacitanceTask

Description

Jython spectrometer related task that reads the bit word for the used

integration capacitance and returns a Double1d array of measured capacitances from a calibration file for each reset interval

API Summary

Jython Syntax
`Double1d cap = getEffectiveCapacitance(Frames inFrame[,CalibrationTree calTree][,EffectiveCapacitance effectiveCapacitance])`

Properties
`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
`CalibrationTree calTree [INPUT, Optional, default=default value: None]`
`EffectiveCapacitance effectiveCapacitance [INPUT, Optional, default=default value: None]`
`Double1d cap [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

Frames inFrame [INPUT, MANDATORY, default=NO default value]
input Frames object

CalibrationTree calTree [INPUT, Optional, default=default value: None]
cal file access

EffectiveCapacitance effectiveCapacitance [INPUT, Optional, default=default value: None]
cal file access

Double1d cap [OUTPUT, MANDATORY, default=NO default value]
returned array containing the effective capacitance

History

- 1.0 20051213 JS initial prototype version

- 1.1 20070302 JS updated for FM CRE
- 1.2 20071123 JS adaption to new cal framework
- 1.3 20080222 JS adapt again to new cal framework

1.110. GetObsOverviewDbTask

Full Name:	herschel.pacs.toolboxes.common.GetObsOverviewDbTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import GetObsOverviewDbTask

Description

Get an overview of observations from the versant Database

API Summary

Jython Syntax
`obslist = getObsOverviewDb([instrument] )`

Property
`String instrument [INPUT, optional, default=default = all]`

API details

Property

`String instrument [INPUT, optional, default=default = all]`

History

- 1.0 04-Sep-2008 EkW The first version is a prototype.

1.111. getPacsCalData

Full Name:	herschel.pacs.cal.GetPacsCalDataTask
Alias:	getPacsCalData
Type:	Java Task -
Import:	from herschel.pacs.cal import GetPacsCalDataTask

Description

Get a PACS calibration product.

This task should be used in every pipeline step for retrieving calibration products. The tasks implements the algorithm to find the correct calibration product based on its arguments.

Example

Example 1: load chopper angle calibration product from calTree

```
cal = getPacsCalData(calTree=calTree, unit="common", type="ChopperAngle")
```

API Summary

Jython Syntax

```
cal = getPacsCalData([cal=myCal][, calTree=calTree]
[, unit="common|spectrometer|photometer"][, type="<className>"]
[, model="FM|QM|FS"][, time=<FineTime>])
```

Properties

- `Product` **`cal`** `[INPUT, OPTIONAL, default=null]`
- `PacsCal` **`calTree`** `[INPUT, OPTIONAL, default=null]`
- `String` **`unit`** `[INPUT, OPTIONAL, default=null]`
- `String` **`type`** `[INPUT, OPTIONAL, default=null]`
- `String` **`model`** `[INPUT, OPTIONAL, default="FM"]`
- `FineTime` **`time`** `[INPUT, OPTIONAL, default=null]`

API details

Properties

`Product cal [INPUT, OPTIONAL, default=null]`

A PACS calibration product which is returned by default when different from `null`.

`PacsCal calTree [INPUT, OPTIONAL, default=null]`

The calibration tree to be used for finding the correct calibration product.

`String unit [INPUT, OPTIONAL, default=null]`

Accepted values are common, spectrometer, photometer (lower case)

String type [INPUT, OPTIONAL, default=null]
The name of the calibration product

String model [INPUT, OPTIONAL, default="FM"]
The instrument model to be used for determining the cal product

FineTime time [INPUT, OPTIONAL, default=null]
The time used to determine the version for time dependent calibration products

1.112. GetPhotHkTask

Full Name:	herschel.pacs.toolboxes.phot.GetPhotHkTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.phot import GetPhotHkTask

Description

unknown

API Summary

Jython Syntax
`photHkTable = getPhotHk(PacketSequence seq)`

Properties
`PacketSequence` **`seq`** `[INPUT, MANDATORY, default=NO default value]`
`TableDatase` **`photHkTable`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`PacketSequence seq [INPUT, MANDATORY, default=NO default value]`
PacketSequence

`TableDatase photHkTable [OUTPUT, MANDATORY, default=NO default value]`
TableDataset with the PHOT HK values

History

- 1.0 23-Oct-2008 EkW initial version

1.113. GetPointingProductTask

Full Name:	herschel.pacs.toolboxes.common.GetPointingProductTask
Type:	Java Task -
Import:	from herschel.pacs.toolboxes.common import GetPointingProductTask
Category:	Task/PACS Task

Description

Retrieves the correct pointing product for a given sequence file or observation context

In case an observation context is provided then the pointing product will be added to this context as well. Either the 'seq' or 'obs' parameter must be provided.

API Summary

Jython Syntax
`pp = getPointingProduct(pool,seq=seq,obs=obs)`

Properties
`ProductStorage; MANDATORY store [INPUT, null, default=no default value]`
`PacketSequence; OPTIONAL seq [INPUT, null, default=no default value]`
`ObservationContext; OPTIONAL obs [INPUT, null, default=no default value]`

API details

Properties

`ProductStorage; MANDATORY store [INPUT, null, default=no default value]`

The product storage

`PacketSequence; OPTIONAL seq [INPUT, null, default=no default value]`

The packetsequence for which this pointing product must be searched

`ObservationContext; OPTIONAL obs [INPUT, null, default=no default value]`

The observation context for which this pointing product must be searched

1.114. getSignalCycleDifference

Full Name:	herschel.pacs.toolboxes.common.getSignalCycleDifference
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import getSignalCycleDifference

Description

Jython function that calculates the average signal values of the flat parts (flat plateaux)

of an input signal cyclecs (only constant rectangular chopping). Then it returns the signal differences and the plateau length.

API Summary

Jython Syntax

```
(Double1d sigDiff, int platLength) =
getSignalCycleDifference(Double1d sigcycle, float stepThreshold,
int lengthTol, float sigmaFactor)
```

Properties
`Double1d` **`sigcycle`** `[INPUT, MANDATORY, default=NO default value]`
`float` **`stepThreshold`** `[INPUT, MANDATORY, default=NO default value]`
`int` **`lengthTol`** `[INPUT, MANDATORY, default=NO default value]`
`float` **`sigmaFactor`** `[INPUT, MANDATORY, default=NO default value]`
`Double1d` **`sigDiff`** `[OUTPUT, MANDATORY, default=NO default value]`
`int` **`platLength`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Double1d sigcycle [INPUT, MANDATORY, default=NO default value]`

array of constant chopper cycles

`float stepThreshold [INPUT, MANDATORY, default=NO default value]`

minimum signal step between chopper plateaux

`int lengthTol [INPUT, MANDATORY, default=NO default value]`

minimum length of a plateau

`float sigmaFactor [INPUT, MANDATORY, default=NO default value]`

factor to standard deviation to define limits of flat plateaux

`Double1d sigDiff [OUTPUT, MANDATORY, default=NO default value]`

signal differences of each plateaux cycle

int platLength [OUTPUT, MANDATORY, default=NO default value]
median length of one chopper plateau

History

- 1.0 20060920 JS initial version
- 1.1 20061005 JS improved version
- 1.2 20070228 JS improved version

1.115. GetSpecResolutionTask

Full Name:	herschel.pacs.toolboxes.spec.GetSpecResolutionTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spec import GetSpecResolutionTask

Description

Jython spectrometer related task that reads the cal file specProperties

and returns the spectral resolution in km/s at a given order and wavelength in micron using the equation described in PICC-ME-SD-004

API Summary

Jython Syntax
`Double1d res = getSpecResolution(int order, Double1d wavelength [,PacsCal calTree][,SpecProperties specProperties][,String model] [,FineTime time])`

Properties
`Integer order [INPUT, MANDATORY, default=No default value]`
`Double1d wavelength [INPUT, MANDATORY, default=No default value]`
`PacsCal calTree [INPUT, Optional, default=default value: None]`
`SpecProperties specProperties [INPUT, Optional, default=default value: None]`
`String model [INPUT, Optional, default=default value: "FM"]`
`FineTime time [INPUT, Optional, default=default value: present date]`
`Double1d res [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

Integer order [INPUT, MANDATORY, default=No default value]
input spectral order

Double1d wavelength [INPUT, MANDATORY, default=No default value]
wavelength in micron

PacsCal calTree [INPUT, Optional, default=default value: None]
cal file access

SpecProperties specProperties [INPUT, Optional, default=default value: None]
cal file access (FM)

String model [INPUT, Optional, default=default value: "FM"]
model name FM, FS or QM

FineTime time [INPUT, Optional, default=default value: present date]
time of measurement

Double1d res [OUTPUT, MANDATORY, default=NO default value]
returned spectral resolution in km/s

History

- 1.0 20080303 JS initial version

1.116. GetSpecSensitivityTask

Full Name:	herschel.pacs.toolboxes.spec.GetSpecSensitivityTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spec import GetSpecSensitivityTask

Description

Jython spectrometer related task that reads the cal file Sensitivity

and returns the rms noise in Jy or W/m^2 depending on given order and wavelength in micron using data of AlPog and N. Geis

API Summary

Jython Syntax
`Double1d noise = getSpecSensitivity(int order, Double1d wavelength [,String contline][,PacsCal calTree][,Sensitivity sensitivity] [,String model][,FineTime time])`

Properties
`int order [INPUT, MANDATORY, default=No default value]`
`Double1d wavelength [INPUT, MANDATORY, default=No default value]`
`String contline [INPUT, Optional, default=default value: "cont"]`
`PacsCal calTree [INPUT, Optional, default=default value: None]`
`Sensitivity sensitivity [INPUT, Optional, default=default value: None]`
`String model [INPUT, Optional, default=default value: "FM"]`
`FineTime time [INPUT, Optional, default=default value: present date]`
`Double1d noise [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`int order [INPUT, MANDATORY, default=No default value]`

order 1 (102 - 221 micron), 2 (71 - 105 micron), 23 (extreme 2) (60 - 73.4 micron), or 3 (55 - 74 micron)

`Double1d wavelength [INPUT, MANDATORY, default=No default value]`

wavelength in micron

`String contline [INPUT, Optional, default=default value: "cont"]`

select between continuum noise with unit Jy or line noise with unit W/m^ ("cont" or "line")

`PacsCal calTree [INPUT, Optional, default=default value: None]`

cal file access

Sensitivity sensitivity [INPUT, Optional, default=default value: None]

cal file access (FM)

String model [INPUT, Optional, default=default value: "FM"]

model name FM, FS or QM

FineTime time [INPUT, Optional, default=default value: present date]

time of measurement

Double1d noise [OUTPUT, MANDATORY, default=NO default value]

returned sensitivity in Jy or W/m^2

History

- 1.0 20080305 JS initial version

1.117. GetTelescopeBackgroundTask

Full Name:	herschel.pacs.toolboxes.spec.GetTelescopeBackgroundTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spec import GetTelescopeBackgroundTask

Description

Jython spectrometer related task that reads the cal file TelescopeBackground

and returns the flux in Jy or W/pix depending on given wavelength in micron using data of HF

API Summary

Jython Syntax
`Double1d flux = getTelescopeBackground(Double1d wavelength [,String unit][,PacsCal calTree][,TelescopeBackground telescopeBackground][,String model][,FineTime time])`

Properties
`Double1d` **`wavelength`** `[INPUT, MANDATORY, default=No default value]`
`String` **`unit`** `[INPUT, Optional, default=default value: "Jy"]`
`PacsCal` **`calTree`** `[INPUT, Optional, default=default value: None]`
`TelescopeBackground` **`telescopeBackground`** `[INPUT, Optional, default=default value: None]`
`String` **`model`** `[INPUT, Optional, default=default value: "FM"]`
`FineTime` **`time`** `[INPUT, Optional, default=default value: present date]`
`Double1d` **`flux`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Double1d wavelength [INPUT, MANDATORY, default=No default value]`
wavelength in micron

`String unit [INPUT, Optional, default=default value: "Jy"]`
select between units Jy or W/pix ("Jy" or "W")

`PacsCal calTree [INPUT, Optional, default=default value: None]`
cal file access

`TelescopeBackground telescopeBackground [INPUT, Optional, default=default value: None]`
cal file access (FM)

`String model [INPUT, Optional, default=default value: "FM"]`
model name FM, FS or QM

FineTime time [INPUT, Optional, default=default value: present date]
time of measurement

Double1d flux [OUTPUT, MANDATORY, default=NO default value]
returned flux in Jy or W/pix

History

- 1.0 20080304 JS initial version

1.118. gratCtrl2tds

Full Name:	herschel.pacs.toolboxes.spec.gratCtrl2tds
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spec import gratCtrl2tds

Description

This function transforms the Long1d array of HK measures

DMC_GRAT_CTRL_ST (208) into a TableDataset containing one Column for each of the bit fields contained in the Grating \ Controller Status HK measures.

API Summary

Jython Syntax
`TableDataset tds = gratCtrl2tds(Long1d array)`

Properties
`Long1d array [INPUT, MANDATORY, default=NO default value]`
`TableDataset tds [OUTPUT, MANDATORY, default=NO default value]`

Miscellaneous

This function is necessary for Diagnostic HK, but UNNECESSARY FOR NOMINAL HK, as the parameters described by each bit (or bit field) contained in DMC_GRAT_CTRL_ST can in this case be accessed directly: tds = nominalHKSequence([DMC_GC_ERROR", "DMC_GC_HOM_PROG"])

API details

Properties

Long1d array [INPUT, MANDATORY, default=NO default value]
input (Diag) HK measures for DMC_GRAT_CTRL_ST (HK par. 208)

TableDataset tds [OUTPUT, MANDATORY, default=NO default value]
one column for each bit field Column 1 is Int1d, all the others, Short1d

History

- 1.0 20050706 PR initial version
- 1.1 20050804 JS header adopted to jtags syntax
- 1.2 20051107 JS clean up of imports

1.119. gratPos2Wave

Full Name:	herschel.pacs.spg.spec.gratPos2Wave
Type:	Jython Task -
Import:	from herschel.pacs.spg.spec import gratPos2Wave

Description

Calculate the wavelength corresponding to a grating position

Evaluates the Littrow equation as described in : "Wavelength Calibration of the PACS Spectrometer AVM/CQM" ILT PTD 4.2.1 FGB, Issue 1.0, feb 4,2005, with a polynomial relation between grating position and alpha per pixel

API Summary

Jython Syntax

```
Double wave = gratPos2Wave(Double gratPos, String band, int pix,
int mod,
[, PacsCal calTree][, LittrowPolynomes littrowPolynomes]
[, model = "FM"][, time=Date()])

Double1d wave = gratPos2Wave(Double1d gratPos, String band, int
pix, int mod,
[, PacsCal calTree][, LittrowPolynomes littrowPolynomes]
[, model = "FM"][, time=Date()])
```

Properties

- Double1d **gratPos** [INPUT, MANDATORY, default=NO default value]
- String **band** [INPUT, MANDATORY, default=NO default value]
- int **pix** [INPUT, MANDATORY, default=NO default value]
- int **mod** [INPUT, MANDATORY, default=NO default value]
- PacsCal **calTree** [INPUT, OPTIONAL, default=None]
- LittrowPolynomes **littrowPolynomes** [INPUT, OPTIONAL, default=None]
- String **model** [INPUT, Optional, default=None]
- FineTime **time** [INPUT, Optional, default=None]
- Double1d **wave** [OUTPUT, MANDATORY, default=NO default value]

API details

Properties

Double1d gratPos [INPUT, MANDATORY, default=NO default value]

Grating position readback

String band [INPUT, MANDATORY, default=NO default value]

Spectral band: B3A : Blue short wave channel : filter A, order=3, 50-70um B2A : Blue short wave channel : filter A, order=2, 50-70um B2B : Blue long wave channel : filter B, order=2, 70-100um R1 : Red channel : order=1, 100-200um

int pix [INPUT, MANDATORY, default=NO default value]

Pixel number within a module (1,2,3 .., 16)

int mod [INPUT, MANDATORY, default=NO default value]

Module number (0,1,2,3 .., 24)

PacsCal calTree [INPUT, OPTIONAL, default=None]

cal file access

LittrowPolynomes littrowPolynomes [INPUT, OPTIONAL, default=None]

Specific version of LittrowPolynomes calibration table to use

String model [INPUT, Optional, default=None]

model name FM, FS or QM

FineTime time [INPUT, Optional, default=None]

time of measurement

Double1d wave [OUTPUT, MANDATORY, default=NO default value]

computed wavelength in um

History

- 1.0 07-Mar-2005 EkW - Initial version as waveCalc
- Basing on BVs py script and FGBs Littrow equation
- 1.1 24-Mar-2005 EkW - Converted to Toolbox function
- 1.2 27-Apr-2005 BV - Changed interface and read parameters from calfile
- 1.3 04-Aug-2005 JS - header adopted to jtags syntax
- 1.4 18-Mar-2007 BV - Changed default version to FM_1_0
- 2.0 20-Nov-2007 EkW - adopt to new calibration framework
- 2.1 22-Feb 2008 JS - adapt again to new cal framework
- 3.0 18-Mar-2008 BV - Changed to use LittrowPolynomes, merged with 2.1
- 3.1 23-Apr-2009 BV - SPR PACS 1543 - don't read calfile if it is passed as
- an argument already

1.120. header1

Full Name:	herschel.pacs.docfiles.howtos.header1
Type:	Jython Task -
Import:	from herschel.pacs.docfiles.howtos import header1

Description

Function that calculates a percentile clipped mean signal for each pixel

This function first sorts the signal values for each detector pixel. Afterwards, it averages the signals in a timeline after discarding a given percentage of the highest values defined by the input parameter hiReject and a given percentage of the lowest values defined by the input parameter loReject. This is carried out for each pixel and the resulting frame is returned. Additionally, the standard deviation for each pixel is written in the ASCII file stdev.dat.

API Summary

Jython Syntax
`Frames outFrames = cleanMeanSignals(Frames inFrames, Double loReject, Double hiReject [, ascii_out = <ascii_out>])`

Properties
`Frames inFrames [INPUT, MANDATORY, default=NO default value]`
`Double loReject [INPUT, MANDATORY, default=NO default value]`
`Double hiReject [INPUT, MANDATORY, default=NO default value]`
`int ascii_out [INPUT, OPTIONAL, default=0]`
`Frames outFrames [OUTPUT, MANDATORY, default=NO default value]`
`file stdev.dat [OUTPUT, OPTIONAL, default=NO default value]`

Miscellaneous

In future, the code should check if loReject + hiReject >=100. The name of the output ASCII file should be definable by the user.

API details

Properties

Frames inFrames [INPUT, MANDATORY, default=NO default value]
Frames object

Double loReject [INPUT, MANDATORY, default=NO default value]
percentage of lowest values to discard

Double hiReject [INPUT, MANDATORY, default=NO default value]
Percentage of highest values to discard

int ascii_out [INPUT, OPTIONAL, default=0]
Indicates if the stdev.dat file should be written (1) or not (0)

Frames outFrames [OUTPUT, MANDATORY, default=NO default value]
Returned averaged Frames object

file stdev.dat [OUTPUT, OPTIONAL, default=NO default value]
ASCII file that contains the value of the standard deviation for each pixel.

See also

- ???

History

- 1.0 23-Mar-2005 JS Initial version of this Task

1.121. HolderList

Full Name:	herschel.pacs.spg.phot.deglitching.WTMML.util.HolderList
Type:	Java Class -
Import:	from herschel.pacs.spg.phot.deglitching.WTMML.util import HolderList
Category:	class

1.122. HolderRep

Full Name:	herschel.pacs.spg.phot.deglitching.WTMML.util.HolderRep
Type:	Java Class -
Import:	from herschel.pacs.spg.phot.deglitching.WTMML.util import HolderRep

History

- 23 march 2006 : first java version

1.123. IIndLevelDeglitchTask

Full Name:	herschel.pacs.spg.phot.IIndLevelDeglitchTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import IIndLevelDeglitchTask

Description

This task takes a MapIndex and a frames object as input and creates a Map out of the data. The task can either use deglitching or not.

The deglitching algorithm of IIndLevelDeglitchTask looks at all signal contributions to a pixel of the map. References to these contributions are stacked in the MapIndex. A Sigclip algorithm checks outliers in the contributing signals and removes them before co-adding the remaining values to a map. Instead of creating a map, it is an option to flag the outliers that Sigclip has found in a mask. The mask is put into the input frames object.

Since the MapIndex stores references to the map contributions, the frames object that has been used to create the index is a mandatory input.

API Summary

Jython Syntax

`SimpleImage map = IIndLevelDeglitch(MapIndex index, Frames inframes, [, map = True]`
`[, mask = False] [, maskname = "my maskname"]`
`[, submap = Int1d([100, 100, 50, 50])] [, algo = Sigclip(10, 4)])`

Properties
`MapIndex` **`index`** `[INPUT, MANDATORY, default=NO default value]`
`Frames` **`inframes`** `[INPUT, MANDATORY, default=NO default value]`
`Boolean` **`map`** `[INPUT, OPTIONAL, default=default value: True]`
`Boolean` **`mask`** `[INPUT, OPTIONAL, default=default value: True]`
`String` **`maskname`** `[INPUT, OPTIONAL, default=default value: "2nd level glitchmask"]`
`Int1d` **`submap`** `[INPUT, OPTIONAL, default=default value: full mapsize]`
`SimpleImage` **`sourcemask`** `[INPUT, OPTIONAL, default=no default.]`
`Double` **`threshold`** `[INPUT, OPTIONAL, default=default: 0.5.]`
`Sigclip` **`algo`** `[INPUT, OPTIONAL, default=default value: Sigclip(behavior = Sigclip.CLIP).]`
`OPTIONAL` **`weightedsignal INPUT`** `[Boolean, default is False., default=no default value]`
`OPTIONAL` **`deglitchvector INPUT`** `[String, default is "framessignal"., default=no default value]`

API details

Properties

MapIndex index [INPUT, MANDATORY, default=NO default value]

The input Mapindex object. It must have been created with the PhotProjectTask from the inframes object.

Frames inframes [INPUT, MANDATORY, default=NO default value]

The input frames object that has been used to create the MapIndex.

Boolean map [INPUT, OPTIONAL, default=default value: True]

A boolean value. Set it to true (default), if you want the task to create a SimpleImage that contains a map of the inframes/index data. If you require only a mask, map = False will save computing time.

Boolean mask [INPUT, OPTIONAL, default=default value: True]

If true (default), creates a mask from the deglitch result that flags all values of the inframes object, that the input algorithm markes as outliers. The mask will be added to the inframes object. If you require only a map, mask = false will save computing time.

String maskname [INPUT, OPTIONAL, default=default value: "2nd level glitchmask"]

the name of the new mask that is created in the frames object, if mask is true.

Int1d submap [INPUT, OPTIONAL, default=default value: full mapsize]

specifies the geometry of a submask. This can be used to quickly optimize Sigclip settings before a long computation of the whole map is performed. The format of the values are (in respect to the output map): Int1d({top left row index, top left column index, heigth (= number of rows), width (= number of columns)}).

There is no extra boundary check. If the specified submap overflows the output map dimensions, an exception is thrown during computation.

SimpleImage sourcemask [INPUT, OPTIONAL, default=no default.]

A simple image that is used as a mask. All values > threshold (with threshold being another parameter of this task) will be treated as masked and not deglitched. All other values will be deglitched. The SimpleImage must have the same dimensions as the final map. Recommended is to create a map first and use this map as input to mask sources in a second run.

Double threshold [INPUT, OPTIONAL, default=default: 0.5.]

A threshold value that determines the masked values of the parameter sourcemask. All values > threshold will be treated as masked and not deglitched. All other values will be deglitched.

Sigclip algo [INPUT, OPTIONAL, default=default value: Sigclip(behavior = Sigclip.CLIP).]

A Sigclip algorithm with user specified settings that is used for deglitching.

OPTIONAL weightedsignal INPUT [Boolean, default is False., default=no default value]

set this to true, if the signal contributions to the map pixels should be weighted with the signal error. The error is taken from the noise array in the inframes object.

OPTIONAL deglitchvector INPUT [String, default is "framessignal"., default=no default value]

This parameter specifies, which values are used for the deglitching: deglitchvector = "framessignal": for every mappixel the list of signal contributions is extracted from the input frames object. Sigma clipping is performed on this list. deglitchvector = "timeordered": all signal contributions that belong to a detector array (= that have the same time index) are multiplied with their weight factors (poids/pixelsurface) and coadded. The list of these weighted signal contributions is sigma clipped. In case of a positive clipping result, all contributions of a timeframes are excluded from the map.

History

- 1.0 20090721 MW initial version

1.124. join

Full Name:	herschel.pacs.toolboxes.numeric.join
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import join

Description

2 Double2d arrays are joined on reference columns

API Summary

Jython Syntax
`Double2d outdata = join(Double2d in1, Double2d in2, int refcol1=0, int refcol2=, int union=0)`

Properties
`Double2d` **`in1`** `[INPUT, MANDATORY, default=NO default value]`
`Double2d` **`in2`** `[INPUT, MANDATORY, default=NO default value]`
`int` **`refcol1`** `[INPUT, OPTIONAL, default=0]`
`int` **`refcol2`** `[INPUT, OPTIONAL, default=0]`
`int` **`union`** `[INPUT, OPTIONAL, default=0]`
`Double2d` **`outdata`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Double2d in1 [INPUT, MANDATORY, default=NO default value]`
first input array

`Double2d in2 [INPUT, MANDATORY, default=NO default value]`
second input array to be appended

`int refcol1 [INPUT, OPTIONAL, default=0]`
column number of first array on which the match should be established (identifier column)

`int refcol2 [INPUT, OPTIONAL, default=0]`
column number of second array on which the match should be established (identifier column)

`int union [INPUT, OPTIONAL, default=0]`
0 : the output is the intersection of in1 & in2 1 : the output is the union of in1 & in2. The missing data are filed with 0's

`Double2d outdata [OUTPUT, MANDATORY, default=NO default value]`
resulting Double2d array

History

- 20060829 PR 1.0 initial version

1.125. LinearRegression

Full Name:	herschel.pacs.spg.phot.deglitching.WTMML.util.LinearRegression
Type:	Java Class -
Import:	from herschel.pacs.spg.phot.deglitching.WTMML.util import LinearRegression

History

- 27 october 2006 : first java version

1.126. LorentzModel

Full Name:	herschel.pacs.share.numeric.LorentzModel
Type:	Java Class -
Import:	from herschel.pacs.share.numeric import LorentzModel

Description

LorentzModel.

nonlinear model of a Lorentz distribution extends class NonLinearModel, to be placed in package herschel.ia.numeric.toolbox.fit

$f(x:p) = = p_2*1/pi * p_1/2 * 1/((x-p_0)^2 + (p_1/2)^2)$

p_0 = center of the Lorentzian curve p_1 = gamma: half-width factor of Lorenzian curve p_2 = amplitude

The parameters are initialized at {0.0, 1.0, 1.0}.

History

- 1.0 20060907 JS first version
- 1.1 20070111 JS jtags header added

1.127. madmapInvnttUtils

Full Name:	herschel.pacs.toolboxes.spg.madmapInvnttUtils
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spg import madmapInvnttUtils

History

- 1.0 23-Nov-2009 BA Initial version

1.128. makeCalChopperAngle_FS1

Full Name:	herschel.pacs.cal.scripts.makeCalChopperAngle_FS1
Type:	Jython Task -
Import:	from herschel.pacs.cal.scripts import makeCalChopperAngle_FS1

Description

Create a ChopperAngle calibration product of FS chopper based on PICC-MA-TR-068 from Markus Nielbock

Create a dataset with the values for the chopper readout - angle calibration divide the chopper deflection range in 3 different ranges to get better fits of good quality Structure : cal - Zeissfile: file name of Zeiss chopper calibration -String - Voltages : ArrayDataset - Double1d (Unit: Volts) - Angles : ArrayDataset - Double1d (Unit: degrees) - ZeissAmplification : DoubleParameter - CSLAmplification : DoubleParameter - ZeroPointOffset : DoubleParameter (unit: DecMec readouts) - Adu2VoltConversion: DoubleParameter Angle units in degrees, voltage units in DECMEC readouts - Ranges: 3 deflection angle ranges of polynomial fits (Science, Calibration windows) - Double1d (unit: degrees) - PolCalNegVoltAngle: Polynomial coefficients of calibration window negative angles for conversion Voltages to Angle - Double1d (5 elements) - PolCalPosVoltAngle: Polynomial coefficients of calibration window positive angles for conversion Voltages to Angle - Double1d (5 elements) - PolScienceVoltAngle: Polynomial coefficients of science window for conversion voltages to angles - Double1d (7 elements) - PolCalNegAngleVolt: Polynomial coefficients of calibration window negative angles for conversion angles to voltages - Double1d (5 elements) - PolCalPosAngleVolt: Polynomial coefficients of calibration window positive angles for conversion angles to voltages - Double1d (5 elements) - PolScienceAngleVolt: Polynomial coefficients of science window for conversion angles to voltages - Double1d (7 elements)

See also

- makeCalChopperAngle_FM_2.py

History

- 25/02/2009 JS - Creation

1.129. makeCalInvnttFromFiles

Full Name:	herschel.pacs.cal.scripts.makeCalInvnttFromFiles
Type:	Jython Task -
Import:	from herschel.pacs.cal.scripts import makeCalInvnttFromFiles

Description

Generate MADMap inverse noise time-time correlation calibration file.

Note: For now, we only have one file, so, use the one file for each of the photometer detectors. Later, pass an array of files, or a file expression to match.

Example

Example 1: makeCalInvnttFromFiles("invntt_2.0", 2048, band="BL", version=1)

```
New Inverse noise time-time correlation calibration file has been created:
PCalPhotometer_InvnttBL_FM_v1.fits
```

History

- 1.0 07-Nov-2007 First version. Make cal files from Dave Frayer's binary invtt files.
- 1.4 21-May-2008 Modify to accomadate all three bands: BL, BS, RED

1.130. makeCalPhotCorrZeroLevelFromFiles

Full Name:	herschel.pacs.cal.scripts.makeCalPhotCorrZeroLevelFromFiles
Type:	Jython Task -
Import:	from herschel.pacs.cal.scripts import makeCalPhotCorrZeroLevelFromFiles

Description

Generate zero-level corr

See also

- ???

History

- 1.0 12-Feb-2007 First version. Make cal files from Dave Frayer's blue and red files.
- 1.4 10-Dec-2008 Add green field and clean up

1.131. makeOBCPDescription

Full Name:	herschel.pacs.cal.scripts.makeOBCPDescription
Type:	Jython Task -
Import:	from herschel.pacs.cal.scripts import makeOBCPDescription

Description

Create a OBCP number to textual description calfile

- OBCPNumber : IntParameter - DMCSNumber : IntParameter - OBCPDescription : StringParameter - DMCSDescription : StringParameter

History

- 14/02/2006 EkW - Creation following TMs input

1.132. MakePacsPointingProductTask

Full Name:	herschel.pacs.spg.MakePacsPointingProductTask
Type:	Java Task -
Import:	from herschel.pacs.spg import MakePacsPointingProductTask
Category:	Task/PACS Task

Description

unknown

API Summary

Jython Syntax
`pacsPointing = makePointingProduct(pp)`

Property
`PointingProduct; MANDATORY pp [INPUT, null, default=no default value]`

API details

Property

`PointingProduct; MANDATORY pp [INPUT, null, default=no default value]`
PointingProduct

1.133. MakePointingProductTask

Full Name:	herschel.pacs.spg.MakePointingProductTask
Type:	Jython Task -
Import:	from herschel.pacs.spg import MakePointingProductTask

Description

Make a PointingProduct from XY Stage coordinates

API Summary

Jython Syntax
`pointingProduct = makePointingProduct([xyTable] [seq] )`

Properties
`String camera [INPUT, MANDATORY, default=no default value]`
`TableDataset xyTable [INPUT, Optional, default=NO default value]`
`PacketSequence seq [INPUT, Optional, default=No default value]`
`PointingProduct pointingProduct [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`String camera [INPUT, MANDATORY, default=no default value]`

Camera identifier : "Blue Photometer", "Red Photometer", "Blue Spectrometer", "Red Spectrometer"

`TableDataset xyTable [INPUT, Optional, default=NO default value]`

TableDataset with XY coordinates (e.g. from PAL)

`PacketSequence seq [INPUT, Optional, default=No default value]`

PacketSequence with XY Stage data

`PointingProduct pointingProduct [OUTPUT, MANDATORY, default=NO default value]`

PointingProduct

History

- 0.5 23-Jun-2008 EkW initial version
- 0.6 18-Aug-2008 EkW in case of no XY stage data , make a staring product (SPR 978)
- 0.7 21-Aug-2008 EkW SPR 975
- 0.8 02-Sep-2008 EkW/JS correct Ra treatment

1.134. MakePrivateNonLinearModel

Full Name:	herschel.pacs.toolboxes.numeric.MakePrivateNonLinearModel
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import MakePrivateNonLinearModel

Description

Jython class that initializes a NonLinearModel and overwrites the result with a given arbitrary function. Define your private function always as def function(input, parameters): ....

Then you can fit this by e.g. a LevenbergMarquard fitter.

Example

Example 1: :

```
define your private function:
def f(x, par):
   return par[0]*exp(x*x)+par[1]*x*x*x+50
Then call the model class by its constructor:
model = MakePrivateNonLinearModel(f, 2)
# initial guesses for the parameters
model.setParameters(Double1d([ 5, 10]))
t = Double1d.range(128)
fitter = LevenbergMarquardtFitter(t,model)
#detsignal is the array[128] to be fitted by the model
param = fitter.fit( detsignal -)
result = model.result(t, param)
p = PlotXY(t, detsignal)
p.addLayer(LayerXY(t,result))
```

API Summary

Jython Syntax
`NonLinearModel model = MakePrivateNonLinearModel(function, int n_parameters)`

Properties
`def function [INPUT, MANDATORY, default=NO default value]`
`int n_parameters [INPUT, MANDATORY, default=NO default value]`

API details

Properties

`def function [INPUT, MANDATORY, default=NO default value]`

input private function

`int n_parameters [INPUT, MANDATORY, default=NO default value]`

number of free parameters of function

History

- 1.0 20060908 JS initial version

1.135. MakeTodArray

Full Name:	herschel.pacs.spg.phot.MakeTodArray
Type:	Java Class -
Import:	from herschel.pacs.spg.phot import MakeTodArray

Description

MakeTodArray

This class provides methods to construct the tod array (a byte array) from Pacs Frames. It is based on David Frayer's write up on "PACS Implementation of MADmap into HCSS system".

API Summary

Constructor
MakeTodArray() MakeTodArray

Method
PacsTodProduct **makeTodArray**(Frames frames, Double scale, Double crota2, Boolean optimizeOrientation, Double minRotation, Boolean chunkScanLegs, Boolean checkTod) makeTodArray

API details

Constructor

MakeTodArray()
MakeTodArray Default constructor

Method

PacsTodProduct makeTodArray(Frames frames, Double scale, Double crota2, Boolean optimizeOrientation, Double minRotation, Boolean chunkScanLegs, Boolean checkTod)
makeTodArray Wrap the tod byte array into a Pacs Product. **Arguments** Frames **frames** [INPUT, MANDATORY, default=no default value] Pacs frames Double **scale** [INPUT, OPTIONAL, default=1.0] Output resolution scaling (as percentage) Double **crota2** [INPUT, OPTIONAL, default=0.0]

PacsTodProduct makeTodArray(Frames frames, Double scale, Double crota2, Boolean optimizeOrientation, Double minRotation, Boolean chunkScanLegs, Boolean checkTod)

Final map rotation

Boolean **optimizeOrientation** [INPUT, OPTIONAL, default=false]

Use automatic map rotation (this will only happen if the calculated rotation angle > minRotation)

Double **minRotation** [INPUT, OPTIONAL, default=15.0]

User defined minimum rotation angle if optimizeOrientation=true

Boolean **chunkScanLegs** [INPUT, OPTIONAL, default=true]

By default, on-target flags are used to chunk scan legs. But user can choose not to do so.

Boolean **checkTod** [INPUT, OPTIONAL, default=false]

Validate tod data if set to true. This is a very slow process! Expert mode only.

Return

PacsTodProduct

Contains tod product and its pertinent information

Errors

Exception

Something went wrong

History

- 2007-30-10 - CL: First implementation, v0.5
- 2007-15-11 - EkW: Include all the changes resulting from testing against Simulator data, v1.0
- 2007-07-12 - JJ: Implement BA's madmap_init.py changes, v1.0
- 2008-29-01 - EkW: Adopt Ra Dec handling / implement writeTod / add pixel size for red array, v1.1
- 2008-29-01 - CL: Use (Ra, Dec) cubes from Frames, v1.1
- 2008-20-02 - CL: Add temporary files clean up before creating tod file, v1.1
- 2008-20-03 - CL: Speed up getTodSkyIndex(), fix a buy in getTodWeights(), v1.27
- 2008-24-03 - CL: Clean up, v1.28
- 2008-26-03 - CL: More clean up, add temporary margin to CRVAL1 & CRCAL2 to accommodate an unknown problem from upstream ILT data, v1.29
- 2008-04-04 - CL: Fixed the vertical orientation bug, v1.30
- 2008-04-10 - CL: Rework to use WCSTransform correctly, v1.31
- 2008-04-17 - CL: Clean up and add more masks handling, v1.32
- 2008-04-29 - EkW: SPR 862 , treatment of Masks, v1.33
- 2008-05-07 - CL: SPR862, reverse EkW's temporary fix of 'treatment of Masks', v1.34
- 2008-05-12 - CL: Get wavelength band information from Frames, v1.35

- 2008-05-16 - CL: Remove 'medianSub' input argument since it is not implemented, v1.36
- 2008-07-22 - CL: Remove tod byte array from memory, instead, write it out as it is being built. Add writing out tod from & to index, v1.39
- 2008-09-24 - CL: Replace Jsky's WCSTransform with herschel.ia.dataset.image.wcs.Wcs. Change TOD index reference point (0,0) from L-L corner to U-L corner to conform herschel.ia.toolbox.mapper updates, v1.40
- 2008-10-31 - CL: Chunking by scan legs using OnTarget flag in Status, v1.42

1.136. MakeTodArrayTask

Full Name:	herschel.pacs.spg.phot.MakeTodArrayTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import MakeTodArrayTask

Description

MakeTodArrayTask

This class is the task that runs makeTodArray(). It calls methods in class MakeTodArray in order to create tod (time-ordered-data) array.

Example

Example 1: makeTodArray

```
todprod = makeTodArray(frames, scale, crota2, optimizeOrientation)
todprod = makeTodArray(frames, optimizeOrientation=true)
todprod = makeTodArray(frames, crota2=30.)
todprod = makeTodArray(frames)
```

API Summary

Properties
Frames **frames** [INPUT, MANDATORY, default=no default value]
Double **scale** [INPUT, OPTIONAL, default=1.0]
Double **crota2** [INPUT, OPTIONAL, default=0.0]
Boolean **optimizeOrientation** [INPUT, OPTIONAL, default=false]
Double **minRotation** [INPUT, OPTIONAL, default=15.0]
Boolean **chunkScaLegs** [INPUT, OPTIONAL, default=true]
Boolean **checkTod** [INPUT, OPTIONAL, default=false]
PacsTodProduct **tod** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

Frames frames [INPUT, MANDATORY, default=no default value]

Input Pacs frames

Double scale [INPUT, OPTIONAL, default=1.0]

Output resolution scaling (as percentage)

Double crota2 [INPUT, OPTIONAL, default=0.0]

Final map rotation

Boolean optimizeOrientation [INPUT, OPTIONAL, default=false]

Use automatic map rotation (this will only happen if the calculated rotation angle > minRotation)

Double minRotation [INPUT, OPTIONAL, default=15.0]

User defined minimum rotation angle if optimizeOrientation=true

Boolean chunkScaLegs [INPUT, OPTIONAL, default=true]

By default, on-target flags are used to chunk scan legs. But user can choose not to do so.

Boolean checkTod [INPUT, OPTIONAL, default=false]

Check if there are any non-observation sky pixels (extremely slow)

PacsTodProduct tod [OUTPUT, MANDATORY, default=no default value]

Contains tod product and its pertinent information

1.137. MapIndex2signalCubeTask

Full Name:	herschel.pacs.spg.phot.MapIndex2signalCubeTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import MapIndex2signalCubeTask

Description

This task backprojects a MapIndex into a signal cube. In this way it creates the signal cube as it has to be in order to create the map.

This allows comparison between the measured signal and the part of the signal that actually has been used for the map. Please be aware of rounding errors due to the precision of the parameters as stored in the MapIndex.

A backprojected signal cube will always differ slightly from the original input signal cube, even if no values have been masked out.

Rounding errors may vary from map to map, depending on the distribution of the signal into the map. To check the order of magnitude for your map, project and backproject your unmasked data first and note the errors.

Example

Example 1: from hipe:

```
cube = mi2sc(index, frames)
originalsignal = frames.signal
diff = originalsignal -- cube
#map the difference
f = frames.copy()
f.setSignal(diff)
diffmap = photProject(f)
```

API Summary

Properties
MapIndex **index** [INPUT, MANDATORY, default=NO default value]
Frames **inframes** [INPUT, MANDATORY, default=NO default value]
Double3d **cube:** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

MapIndex index [INPUT, MANDATORY, default=NO default value]

The input MapIndex object. It must have been created with the PhotProjectTask or the MapIndexTask from the inframes object.

It is the source for backprojection.

To avoid the calculation completely, use the parameter "image" to submit the prepared map. In that case the submitted image is displayed.

Frames inframes [INPUT, MANDATORY, default=NO default value]

The input frames object## that has been used to create the MapIndex. It is needed to obtain signal values that are not stored in the MapIndex.

Double3d cube: [OUTPUT, MANDATORY, default=no default value]

The backprojected signal cube. It has the same size as the signal cube in the frames object.

1.138. MapIndexTask

Full Name:	herschel.pacs.spg.phot.MapIndexTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import MapIndexTask

Description

This task prepares the second level deglitching by calculating a MapIndex. A Mapindex is an object that - like a final level2 map - contains all information about fluxes at a certain sky position. But as opposed to the map, the MapIndex did not co-add all flux contributions to a sky pixel, it stores them instead as a list. This list can be analysed by second level deglitching to sigma clip excessively large flux contributins. After the clipping, the second level deglitching task will co-add the flux contributions and thus create the final (deglitched) map.

MapIndex comes in 2 flavours: the SlimIndex, which is a memory saving minimal information container, and the FullIndex. The FullIndex contains all necessary data for the deglitching,without extra calculation. Please find more details about the two indexes and how to retrieve their data in the javadoc (developer reference documentation) of herschel.pacs.signal.MapIndex.

The rule of thumb is, if you are short with memory, let this task construct a slimindex by using the parameter slimindex = True (default). With more memory available, slimindex = False will instruct this task to return the FullIndex.

The MapIndex is one of the input paramter of the second level deglitching task. It can, though, also be inspected manually by retrieving and plotting its contents from the commandline.

Example

Example 1: from Jide:

```
from herschel.pacs.spg import MapIndexTask
mapindex = MapindexTask()
midx = mapindex(frames, slimindex=False, optimizeOrientation=True)
<br>
```

API Summary

Properties
MANDATORY **inframes: INPUT** [Frames, no default value., default=no default value]
OPTIONAL **outputPixelsize: INPUT** [DOUBLE, default is the same value as the inframes pixelsize (3.2 arcsecs for the blue photometer, default=6.4 for red).]
OPTIONAL **optimizeOrientation: INPUT** [Boolean, default is False., default=no default value]
OPTIONAL **minRotation: INPUT** [Integer, default value is 15., default=no default value]
OPTIONAL **wcs: INPUT** [Wcs, no default value., default=no default value]
OPTIONAL **calibration: INPUT** [Boolean, default is True., default=no default value]

Properties
OPTIONAL **slimindex: INPUT** [Boolean, default value is True., default=no default value]
OPTIONAL **calTree: INPUT** [PacsCal, default is the system caltree., default=no default value]
OPTIONAL **index: OUTPUT** [MapIndex, no default value., default=no default value]

API details

Properties

MANDATORY inframes: INPUT [Frames, no default value., default=no default value]

the input frames class that contains the data for the final map. The frames object has to be processed to level1.

OPTIONAL outputPixelsize: INPUT [DOUBLE, default is the same value as the inframes pixelsize (3.2 arcsecs for the blue photometer, default=6.4 for red).]

the size of a pixel in the output dataset in arcseconds. Default is the same size as the input (6.4 arcsecs for the red and 3.2 arcsecs for the blue photometer).

OPTIONAL optimizeOrientation: INPUT [Boolean, default is False., default=no default value]

rotates the map by an angle between minRotation and 89 degrees in a way that the coverage of the outputimage is maximized

as a result, north points no longer upwards.

Possible values False (default): no automatic rotation, True: automatic rotation

OPTIONAL minRotation: INPUT [Integer, default value is 15., default=no default value]

defines the minimum angle for which optimizeOrientation should be applied. Default value is 15 degrees. If optimizeOrientation finds out that the angle is below this value, no rotation will be done. A message is logged instead.

OPTIONAL wcs: INPUT [Wcs, no default value., default=no default value]

allows to specify a customized wcs that is used for projection.

Usually MapIndexTask calculates and optimizes its own Wcs. The wcs parameter allows to overwrite the internally created wcs. The easiest way to create a Wcs is to use the Wcs4mapTask (that is also used internally by this task), and modify its parameters.

The necessary paramters to modify are:

wcs.setCrota2(angle) : the rotation angle in decimal degrees (like 45.0 for 45 degrees)

wcs.setCrpix1(crpix1) :The reference pixel position of axis 1. Use the center of your map: mapwidth/2 (the Ra-coordiante)

OPTIONAL wcs: INPUT [Wcs, no default value., default=no default value]

wcs.setCrpix2(crpix2) :The reference pixel position of axis 2. Use the center of your map: mapheight/2 (the Dec-coordinate)

wcs.setCrval1(crval1) :The coordinate of crpix1 (ra-value in decimal degrees - use Maptools.ra_2_decimal(int hours, int arcmin, double arcsec) for conversion)

wcs.setCrval2(crval2) :The coordinate of crpix2 (dec-value in decimal degrees - use Maptools.dec_2_decimal(int degree, int arcmin, double arcsec) for conversion)

wcs.setParameter("NAXIS1", mapwidth, "Number of Pixels along axis 1.") :mapwidth = crpix1*2, if you follow these instructions

wcs.setParameter("NAXIS2", mapheight, "Number of Pixels along axis 2") :mapheight = crpix2*2, if you follow these instructions

OPTIONAL calibration: INPUT [Boolean, default is True., default=no default value]

decides, whether the spatial calibration files should be used to calculate pixel geometries

(calibration = True), or whether the detector pixels are assumed to be quadratic with the textbook sizes (calibration = False). Default value: true.

If false, the frames object must contain Ra/Dec coordinate arrays.

OPTIONAL slimindex: INPUT [Boolean, default value is True., default=no default value]

together with deglitch = True instructs PhotProject to build a memory efficient index for deglitching.

Building an slimindex means that second level deglitching will take longer, but can be processed with significantly less memory requirements (ca. 20% compared to slimindex = false).

OPTIONAL calTree: INPUT [PacsCal, default is the system caltree., default=no default value]

CalibrationTree for calfile access

OPTIONAL index: OUTPUT [MapIndex, no default value., default=no default value]

The MapIndex object. Depending on the optional parameter slimindex, it is either a SlimIndex (slimindex = True), or a FullIndex (slimindex = False).

1.139. MapIndexViewerTask

Full Name:	herschel.pacs.spg.phot.MapIndexViewerTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import MapIndexViewerTask

Description

This task invokes the MapIndex Viewer. The MapIndex contains all data contributions of the frames object to a sky map. The MapIndexViewer is the tool to plot and inspect this data.

This task accepts three paramteres. Its the users responsibility to ensure, that the submitted frames object has been used to create the mapindex and the corresponding map. If the three paramters don't fit together, unforeseeable results may be obtained.

Example

Example 1: from hipe:

```
#the import is done during hipe startup
#from herschel.pacs.spg.phot import MapIndexViewerTask
#mapIndexViewer = MapIndexViewerTask()
map = photProject(frames)
mapindex = mapindex(frames)
# alternative calculation for the mapindex:
# photProject(frames, deglitch = True)
# mapindex = photProject.getValue("index")
#
mapIndexViewer(mapindex, frames, map) #invokes the viewer
mapIndexViewer(mapindex, frames) #invokes the viewer with on-the-fly
calculation of the map
```

API Summary

Properties
MapIndex **index** [INPUT, MANDATORY, default=NO default value]
Frames **inframes** [INPUT, MANDATORY, default=NO default value]
SimpleImage **image:** [INPUT, OPTIONAL, default=NO default value]

API details

Properties

MapIndex index [INPUT, MANDATORY, default=NO default value]

The input MapIndex object. It must have been created with the PhotProjectTask or the MapIndexTask from the inframes object.

The MapIndex will be used to visualize the map by in-situ calculation. This may take a while, depending on the size of the map. The calculation is quicker, if the MapIndex is a FullMapIndex, and not a SlimMapIndex.

To avoid the calculation completely, use the parameter "image" to submit the prepared map. In that case the submitted image is displayed.

Frames inframes [INPUT, MANDATORY, default=NO default value]

The input frames object## that has been used to create the MapIndex. It is needed to obtain signal values that are not stored in the MapIndex.

SimpleImage image: [INPUT, OPTIONAL, default=NO default value]

A SimpleImage that contains the map. It is mandatory that the map has been constructed from the input frames object. Although this arameter is optional, it shortcuts the initial on-the-fly map construction from the mapindex.

1.140. meanPassband

Full Name:	herschel.pacs.toolboxes.numeric.meanPassband
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import meanPassband

Description

Jython function to calculate the mean y value in xy-series over x

passband (from a min to a max abscissae position)

API Summary

Jython Syntax
`meanPassband(Double1d abscissae, Double1d inArray [,min = <min> [,max = <max>])`

Properties
`Double1d abscissae [INPUT, MANDATORY, default=NO default value]`
`Double1d inarray [INPUT, MANDATORY, default=NO default value]`

Miscellaneous

- no proper type and rank checking done in all functions

API details

Properties

`Double1d abscissae [INPUT, MANDATORY, default=NO default value]`
abscissa x- values

`Double1d inarray [INPUT, MANDATORY, default=NO default value]`
values of y-axis, number must be equal to number of abscissae values

History

- `2006-11-04 - SG:` - First version
- `2007-07-10 - SG:` - new version
- `2007-09-27 - SG:` - put it into a toolbox

1.141. MergeFramesHkTask

Full Name:	herschel.pacs.toolboxes.common.MergeFramesHkTask
Type:	Java Task -
Import:	from herschel.pacs.toolboxes.common import MergeFramesHkTask

Description

Task : Merge Housekeeping data to Frames (Interpolate, if possible)

task that adds the measures of a housekeeping parameter as frames status field

API Summary

Jython Syntax

```
Frames outFrames = mergeFramesHk(Frames inFrame, PacketSequence
seq, String hkParName [,Boolean raw=raw] [,int
copy=copy] [,String[] hkNames=hkNames] [,Boolean noInter =
noInter] )
```

Properties

- Frames **inFrame** [INPUT, MANDATORY, default=NO default value]
- PacketSequence **hkData** [INPUT, MANDATORY, default=NO default value]
- String **hkParName** [INPUT, MANDATORY, default=NO default value]
- Boolean **raw** [INPUT, Optional, default=default value: False]
- Integer **copy** [INPUT, Optional, default=default value: 0]
- array(String) **hkNames** [INPUT, Optional, default=default value: 0]
- Boolean **noInter** [INPUT, Optional, default=default value: False]

API details

Properties

Frames inFrame [INPUT, MANDATORY, default=NO default value]

input Frames object

PacketSequence hkData [INPUT, MANDATORY, default=NO default value]

input house keeping data as PacketSequence

String hkParName [INPUT, MANDATORY, default=NO default value]

name of status field

Boolean raw [INPUT, Optional, default=default value: False]

indicates if converted/engineering HK values (False), or raw HK values (True) are copied

Integer copy [INPUT, Optional, default=default value: 0]

indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

array(String) hkNames [INPUT, Optional, default=default value: 0]
if an additional list of HK parameters should be added to the Frames

Boolean noInter [INPUT, Optional, default=default value: False]
True : No Interpolation, default : False = with interpolation

History

- 1.0 20060119 JS initial version
- 1.1 20060717 JS debugged and improved
- 1.2 20060724 JS fix: last index considered
- 1.3 20060829 JS extended for all HK data types (including String, Boolean)
- 1.4 20060904 JS fix of String casting
- 1.5 20060906 JS cubic spline interpolation introduced for Int1d, Long1d, Float1d, Double1d
- HK data types, String1d, Bool1d uses closest time values as before
- 1.6 20060907 JS parameter raw added which indicates wether HK values should be read as raw or converted
- 1.7 20060913 JS catch empty HK parameters and return input frames
- 1.8 20060922 JS switch from CubicSplineInterpolator to LinearInterpolator due to severe extrapolation problems
- -> very big deviations of extrapolation results.
- 1.9 20060925 JS expanded interface for HK String array as input for several HK parameters at once
- 1.10 20060928 JS System imported
- 1.11 20061116 EkW converted to java
- 1.12 20061214 EkW add no interpolation keyword
- 1.13 20070111 JS add jtags header

1.142. MergeFramesTableTask

Full Name:	herschel.pacs.spg.MergeFramesTableTask
Type:	Java Task -
Import:	from herschel.pacs.spg import MergeFramesTableTask

Description

Task based on MergeFramesHk: Merge Table data to Frames Status (Interpolate, if possible)

The Table need to have the first Column Time in microseconds since .... (TAI)

API Summary

Jython Syntax
`Frames outFrames = mergeFramesTable(Frames inFrame, TableDataset table [,Boolean noInter=noInter]`
`[,Integer copy=copy] )`

Properties
`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
`TableDataset table [INPUT, MANDATORY, default=NO default value]`
`Integer copy [INPUT, Optional, default=default value: 0]`
`Boolean noInter [INPUT, Optional, default=default value: False]`

API details

Properties

Frames inFrame [INPUT, MANDATORY, default=NO default value]
input Frames object

TableDataset table [INPUT, MANDATORY, default=NO default value]
input table to be merged

Integer copy [INPUT, Optional, default=default value: 0]
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

Boolean noInter [INPUT, Optional, default=default value: False]
True : No Interpolation, default : False = with interpolation

History

- 1.0 20061208 EkW initial version
- 1.1 20070111 JS jtags header added

1.143. MMTDeglitchingTask

Full Name:	herschel.pacs.spg.phot.MMTDeglitchingTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import MMTDeglitchingTask

Description

Deglitching algorithm using the Multiresolution Median Transform as described by Stark et. al.

The algorithm carries out the following steps:

- calculate the wavelet transform of the inFrames signal (= image) cube using a median filter algorithm

- estimate the noise in the image. Used methods are either an average method described in (3), or an iterative refinement described in (2)

- use the image noise to calculate the noise in the wavelet space of the image and eliminate all values in that space larger than nsigma*(wavelet noise)

Literature:

(1) Isocam Data Processing, Stark, Abergel, Aussel, Sauvage, Gastaud et. al., Astron. Astrophys. Suppl. Ser. 134, 135-148 (1999)

(2) Automatic Noise Estimation from the Multiresolution Support, Starck, Murtagh, PASP, 110, 193-199 (1998)

(3) Estimation of Noise in Images: An Evaluation, Olsen, Graphical Models and Image Processing, 55, 319-323 (1993)

API Summary

Jython Syntax
`outframes = mMTDeglitching(inFrames [,copy=copy] [,scales=scales] [,mmt_startenv=mmt_startenv] [,incr/fact=incr/fact] [,mmt_mode=mmt_mode]`
`[,mmt_scales=mmt_scales] [,nsigma=nsigma])`

Properties
`type` **`outFrames : the returned Frames object`** `[INPUT, MANDATORY, default=no default value]`
`type` **`inFrames : the Frames object with the data that should be deglitched`** `[INPUT, MANDATORY, default=no default value]`
`type` **`copy : boolean. Possible values: false (jython: 0) - inFrames will be modified and returned`** `[INPUT, MANDATORY, default=no default value]`
`type` **`scales : int. Number of wavelet scales. This should reflect the maximum expected readout number of the glitches. Default is 5 readouts.`** `[INPUT, MANDATORY, default=no default value]`
`type` **`mmt_startenv : int. The startsize of the environment box for the median transform. Default is 1 readout (plus/minus).`** `[INPUT, MANDATORY, default=no default value]`

Properties
type **incr_fact : float. Increment resp. factor to enhance the mmt_startenv. Default is 1 for mmt_mode == "add" and 2 for mmt_mode == "multiply".** [INPUT, MANDATORY, default=no default value]
type **mmt_mode : String. Defines how the environment should be modified between the scales. Possible values: "add" or "multiply". Default is "add".** [INPUT, MANDATORY, default=no default value]
type **noiseDetectionLimit : double. Threshold for determining the image noise. values between 0.0 and 1.0. Default is 0.3.** [INPUT, MANDATORY, default=no default value]
type **nsigma : int. Limit that defines the glitches on the wavelet level. Every value larger than nsigma*sigma will be treated as glitch. Default is 5.** [INPUT, MANDATORY, default=no default value]
type **onlyMask : boolean. If set to true** [the deglitching will only create a glitchmaks and not remove the glitches from the signal., MANDATORY, default=no default value]
type **maskname : this paramter allows to set a custom maskname for the glitchmask** [that is added to the frames object by this task.., MANDATORY, default=no default value]
type **sourcemask : mmt deglitching is not only sensitive to glitches but also to pointsources. To avoid deglitching of sources** [the sourcemask may mask the locations of sources., MANDATORY, default=no default value]
the standard deviation will only be calculated from the unmasked samples. **use_masks : this paramter determines** [whether the masks are used to calculate the noise. If set to true, MANDATORY, default=no default value]
type **noiseByMeanFilter : this paramter has effect** [if neither a noise array or a noise value is submitted by the user. In that case, MANDATORY, default=no default value]
type **noiseModel: a Double1d that models the noise. The standard internal approach is to use a Gaussian noise with a standard deviation of 1.** [INPUT, MANDATORY, default=no default value]
type **imagenoise a Double with the noise in the image (if it is the same for every image pixel)** [INPUT, MANDATORY, default=no default value]

API details

Properties

type outFrames : the returned Frames object [INPUT, MANDATORY, default=no default value]

type inFrames : the Frames object with the data that should be deglitched [INPUT, MANDATORY, default=no default value]

type copy : boolean. Possible values: false (jython: 0) - inFrames will be modified and returned [INPUT, MANDATORY, default=no default value]

true (jython: 1) - a copy of inFrames will be returned

type scales : int. Number of wavelet scales. This should reflect the maximum expected readout number of the glitches. Default is 5 readouts. [INPUT, MANDATORY, default=no default value]

type mmt_startenv : int. The startsize of the environment box for the median transform. Default is 1 readout (plus/minus). [INPUT, MANDATORY, default=no default value]

type incr_fact : float. Increment resp. factor to enhance the mmt_startenv. Default is 1 for mmt_mode == "add" and 2 for mmt_mode == "multiply". [INPUT, MANDATORY, default=no default value]

type mmt_mode : String. Defines how the environment should be modified between the scales. Possible values: "add" or "multiply". Default is "add". [INPUT, MANDATORY, default=no default value]

example:

the environmentsize for the subsequent median transform environment boxes will be

env(0) = mmt_startenv, env(n) = env(n-1) mmt_mode incr/fact

default for mode "add" means then: env(0) = 1, env(1) = 1 + 1, env(2) = 2 + 1 etc.

default for mode "multiply" means: env(0) = 1, env(1) = 1*2, env(2) = 2*2 env(3) = 4*2 etc.

type noiseDetectionLimit : double. Threshold for determining the image noise. values between 0.0 and 1.0. Default is 0.3. [INPUT, MANDATORY, default=no default value]

type nsigma : int. Limit that defines the glitches on the wavelet level. Every value larger than nsigma*sigma will be treated as glitch. Default is 5. [INPUT, MANDATORY, default=no default value]

type onlyMask : boolean. If set to true [the deglitching will only create a glitchmaks and not remove the glitches from the signal., MANDATORY, default=no default value]

If false, the glitchmask will be created and the detected glitches will be removed from the signal. Default value: true

type maskname : this paramter allows to set a custom maskname for the glitchmask [that is added to the frames object by this task.., MANDATORY, default=no default value]

Default value: MMT_Glitchmask

type sourcemask : mmt deglitching is not only sensitive to glitches but also to pointsources. To avoid deglitching of sources [the sourcemask may mask the locations of sources., MANDATORY, default=no default value]

If this mask is provided in the frames object, the masked locations will not be deglitched by this task. After the task is executed, the sourcemask will be deactivated.

sourcemask is the name of a mask submitted as a String.

type sourcemask : mmt deglitching is not only sensitive to glitches but also to pointsources. To avoid deglitching of sources [the sourcemask may mask the locations of sources., MANDATORY, default=no default value]

Use the PhotReadMaskFromImageTask to write a mask into the frames object. Instruction how to do it are provided in the documentation of the task.

Default value: "".

the standard deviation will only be calculated from the unmasked samples. use_masks : this paramter determines [whether the masks are used to calculate the noise. If set to true, MANDATORY, default=no default value]

Default value: false

type noiseByMeanFilter : this paramter has effect [if neither a noise array or a noise value is submitted by the user. In that case, MANDATORY, default=no default value]

MMTDeglitching has to calculate image noise internally. This can be done by simply subtract all timelines by their mean filtered values (noiseByMeanFilter = true).

If noiseByMeanFilter = false (default) the noise will be calculated in the wavelet space as describe by Starck and Murtagh.

In both cases, the noise will be calculated separately for every pixel.

After execution of the task, the noise can be inspected:
MMTDeglitching.getImageNoiseArray() returns the noise array

type noiseModel: a Double1d that models the noise. The standard internal approach is to use a Gaussian noise with a standard deviation of 1. [INPUT, MANDATORY, default=no default value]

This value is needed to calculate the noise in wavelet space (see (3)). Default: a Gaussian noise created by the class herschel.pacs.share.math.GaussianNoise with the standard deviation of 1. The length of the default data is 100.000.

Parameters that allow diagnostic insight into the algorithm

These paramters are by default calculated by the algorithm without user interaction. If they are set externally, the algorithm will skip the calculation and use the user provided values.

This way it is for example possible to run the MMTDeglitching task once, get out the intermediate results for a check, and feed in corrected or better values to improve the results in further runs.

type imagenoise a Double with the noise in the image (if it is the same for every image pixel) [INPUT, MANDATORY, default=no default value]

History

- 06-07-2007 MW: initial Version

- 30-01-2008 MW: added external noise input
- 15-07-2008 MW: improved memory consumption
- 18-02-2009 MW: deglitching adds a mask
- 07-04-2009 MW: noise per pixel
- 02-07-2009 MW: improved noise detection algorithm (SPR1497)

1.144. Modules2IntegralField

Full Name:	herschel.pacs.toolboxes.spec.Modules2IntegralField
Type:	Java Class -
Import:	from herschel.pacs.toolboxes.spec import Modules2IntegralField

Description

Help taken fromn Modules2IntegralFieldTask

API Summary

Properties
Frames **inFrames** [INPUT, MANDATORY, default=NO default value]
Integer **module** [INPUT, OPTIONAL, default=default value : -1]
Integer **startFrame** [INPUT, OPTIONAL, default=default value : -1]
Integer **endFrame** [INPUT, OPTIONAL, default=default value : -1]
Integer **averageWaves** [INPUT, OPTIONAL, default=default value : 0]
Integer **copy** [INPUT, OPTIONAL, default=default value : 1]
Frames **frame** [OUTPUT, MANDATORY, default=NO default value]

API details

Properties

Frames inFrames [INPUT, MANDATORY, default=NO default value]

input Frames object

Integer module [INPUT, OPTIONAL, default=default value : -1]

select row (module) number

Integer startFrame [INPUT, OPTIONAL, default=default value : -1]

select starting frame in timeline

Integer endFrame [INPUT, OPTIONAL, default=default value : -1]

select finishing frame in timeline

Integer averageWaves [INPUT, OPTIONAL, default=default value : 0]

If 1, the time cube is rearranged to be 5x5 instead of 18x25. all the wavelengths, modules between 1 and 16, are averaged.

Integer copy [INPUT, OPTIONAL, default=default value : 1]

Copy the input (1) and generate new output or overwrite the input (0)

Frames frame [OUTPUT, MANDATORY, default=NO default value]

returned frame with 5x5 cube representing integral field view.

1.145. MpeModuleReaderT

Full Name:	herschel.pacs.spg.MpeModuleReaderT
Type:	Java Class -
Import:	from herschel.pacs.spg import MpeModuleReaderT

Description

Reader for Cold Readout Electronic Ramps files.

This ramps represent the raw samples data as they come from the CRE.

1.146. MpiaModuleReader

Full Name:	herschel.pacs.toolboxes.spg.MpiaModuleReader
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spg import MpiaModuleReader

Description

Jython function to convert the CRE-data files of MPIA to a Ramps object

This function converts the selected CRE-data files of one detector module or a 6 pack of MPIA (*.dat, *.sts) to a Ramps Object used in IA for raw data. Some file header information is stored as MetaData. Returns a Ramps object

API Summary

Jython Syntax

```
readObject = MpiaModuleReader(String filename [, int sixpack = <sixpack>])
Ramps outramp = readObject.read()
```

Properties

`String filename [INPUT, MANDATORY, default=NO default value]`

`int sixpack [INPUT, OPTIONAL, default=0]`

`Ramps outramp [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`String filename [INPUT, MANDATORY, default=NO default value]`

file name of one data MPIA data set with extension *.dat

`int sixpack [INPUT, OPTIONAL, default=0]`

optional parameter that indicates if one module or 6 pack data is used 0: default: one module is used 1 or something else: 6 pack data is used

`Ramps outramp [OUTPUT, MANDATORY, default=NO default value]`

MpiaModuleReaderTask readObject: returned instantiated class Ramps outramp: returned filled Ramps object

History

- 1.0 20050613 JS initial version
- 1.1 20050805 JS header adopted to jtags syntax
- 1.2 20051107 JS clean up of imports
- 1.3 20060306 JS setRampValues() -> setSignal()

1.147. MultiresolutionMedianTransform

Full Name:	herschel.pacs.share.numeric.MultiresolutionMedianTransform
Type:	Java Class -
Import:	from herschel.pacs.share.numeric import MultiresolutionMedianTransform

History

- 06-07-2007 MW: initial Version

1.148. normalise

Full Name:	herschel.pacs.toolboxes.numeric.normalise
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import normalise

Description

Jython function to normalize the y values in an xy-series to passband mean/surface

API Summary

Jython Syntax
`normalize(Double1d abscissae, Double1d inArray [,min = <min> [,max = <max>])`

Properties
`Double1d` **`abscissae`** `[INPUT, MANDATORY, default=NO default value]`
`Double1d` **`inarray`** `[INPUT, MANDATORY, default=NO default value]`

Miscellaneous

- no proper type and rank checking done in all functions

API details

Properties

`Double1d abscissae [INPUT, MANDATORY, default=NO default value]`
abscissa x- values

`Double1d inarray [INPUT, MANDATORY, default=NO default value]`

History

- `2006-11-04 - SG:` - First version
- `2007-07-11 - SG:` - new version
- `2007-09-27 - SG:` - put it into a toolbox

1.149. offMapL1

Full Name:	herschel.pacs.spg.pipeline.ipipe.spec.offMapL1
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.ipipe.spec import offMapL1

History

- 1.0 20090318 JS initial version
- 1.1 20090624 JS updated
- 1.2 20090716 JS introduce slicing

1.150. offMapL2

Full Name:	herschel.pacs.spg.pipeline.ipipe.spec.offMapL2
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.ipipe.spec import offMapL2

History

- 1.0 20090318 JS initial version
- 1.1 20090624 JS updated
- 1.2 20090716 JS introduce slicing

1.151. PacsCube

Full Name:	herschel.pacs.signal.PacsCube
Type:	Java Class -
Import:	from herschel.pacs.signal import PacsCube

Description

Usage and examples.

Examples

Example 1: # Pass frames to PacsCubeBuilder task and get a Product back.

```
cube_blue = specFrames2PacsCube (frame_blue)
# Get the flux and wave from the cube.
flux = cube_blue.flux
wave = cube_blue.wave
# Get the label information too.
lbl = cube_blue.lbl
# Use it to get rid of lbl == 0.
gtz = lbl -!= 0
gtzidx = gtz.where(gtz)
# Plot wave -/ flux values for a module corresponding to x=2 and y=2.
p = PlotXY(wave[gtzidx,2,2],flux[gtzidx,2,2],\
titleText="LBL!=1, module(2,2)")
style = p.style
# Set to dots.
style.symbol = 1
# Turn line off.
style.line = 0
```

Example 2: Plot flux vrs. wave for module (2,2) for upscan chop on and off

```
removing unclean chopped data
 # Pass frames to PacsCubeBuilder task and get a Product back.
 cube_blue = specFrames2PacsCube (frame_blue)
 # Get the flux and wave from the cube.
 flux = cube_blue.flux
 wave = cube_blue.wave
 # Use the LBL information to plot the flux vrs. wave values
 # from the up scan chop on against the same values for the
 # chop off.
 lbl = cube_blue.lbl
 uchp = cube_blue.uncleanChop
 on = (lbl == 3) & (uchp == 0)
 off = (lbl == 5) & (uchp == 0)
 idon = on.where(on)
 idoff = off.where(off)
 p1 = PlotXY(wave[idon,2,2],flux[idon,2,2],\
 titleText="Upscan chop on/off, module(2,2), unclean chop removed")
 style = p1.style
 # Set to dots.
 style.symbol = 1
 # Turn line off
 style.line = 0
 p1[1] = LayerXY(wave[idoff,2,2],flux[idoff,2,2])
 style = p1[1].style
 # Set to dots
 style.symbol = 1
 # Turn line off
 style.line = 0
```

Example 3: Plot the individual spectral pixels from PacsCube.

```
pw = pacscube_blue.wave
pf = pacscube_blue.flux
print pw.dimensions -, pf.dimensions
m = pacscube_blue.wave.dimensions[0] -/16
# Plot the individual spectral pixels from the pacs cube.  They should
# look like the frames, since they're the result of decomposing the cloud.
plot = PlotXY(titleText="Central [2,2] spaxel signal -")
pixels = cube.status["PIXEL"].data
for i in range(16):
     pixIdx = pixels.where(pixels==i)
     plot.addLayer(LayerXY(cube.wave[pixIdx], \
        cube.flux[pixIdx]))
     plot[i].line=0
```

API Summary

Properties
Frames **pacsCalFrames** [INPUT, MANDATORY, default=NO default value]
PacsCube **pacsCube** [OUTPUT, MANDATORY, default=NO default value]

API details

Properties

Frames pacsCalFrames [INPUT, MANDATORY, default=NO default value]

input Frames object

PacsCube pacsCube [OUTPUT, MANDATORY, default=NO default value]

1.152. pacsdefault_pipeline

Full Name:	herschel.pacs.spg.pipeline.oldstyle.pacsdefault.pacsdefault_pipeline
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle.pacsdefault import pacsdefault_pipeline

History

- 1.0 20081110 EkW initial version - SPR 1138

1.153. pacsdefault_pipeline

Full Name:	herschel.pacs.spg.pipeline.pacsdefault.pacsdefault_pipeline
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.pacsdefault import pacsdefault_pipeline

History

- 1.0 20081110 EkW initial version - SPR 1138

1.154. pacsdefault_pipeline

Full Name:	herschel.pacs.toolboxes.spg.pacsdefault.pacsdefault_pipeline
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spg.pacsdefault import pacsdefault_pipeline

History

- 1.0 20081110 EkW initial version - SPR 1138

1.155. PacsInvntt

Full Name:	herschel.pacs.spg.PacsInvntt
Type:	Java Class -
Import:	from herschel.pacs.spg import PacsInvntt

Description

PacsInvntt

Invntt calibration information for MadMap

History

- `2008-01-28 - JJ`: First implementation, v0.5
- `2008-05-12 - CL`: Add a place holder for a new constructor to deal with different wavelength bands, v1.5
- `2008-05-21 - CL`: Support 3 different bands: BL, BS, RED, v1.6
- `2008-10-31 - CL`: Chunk by scan legs using OnTarget flag in Status, v1.9

1.156. PacsLevel0Plugin

Full Name:	herschel.pacs.toolboxes.spg.PacsLevel0Plugin
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spg import PacsLevel0Plugin

Description

Base Script for PACS Level 0 Product generation

Example

Example 1: from herschel.ia.pg import PgPluginManager

```
obsid   = 536871716l
env     = None
plugins = PgPluginManager.getInstance()
plugins.register("urn:level0:pacs",PacsLevel0Plugin())
plugin  = plugins.getPgPlugin("urn:level0:pacs")
plugin.populate(obsid, env, -"pacs_fm_ilt_3@pacs5", -"ekki_test3@pacs1.mpe-
garching.mpg.de")
```

API Summary

Jython Syntax

```
plugin.populate(obsid, env, database, pName [,pType] [,hklimit]
[,sclimit])
```

Properties

long **obsid** [INPUT, MANDATORY, default=NO default value]
String **env** [INPUT, MANDATORY, default=NO default value]
String **database** [INPUT, MANDATORY, default=NO default value]
String **pName** [INPUT, MANDATORY, default=NO default value]
String **pType** [INPUT, OPTIONAL, default=default value = "DBPool"]
long **hklimit** [INPUT, OPTIONAL, default=default value = 10000]
long **sclimit** [INPUT, OPTIONAL, default=default value = 1000]

API details

Properties

long obsid [INPUT, MANDATORY, default=NO default value]

Obervation ID

String env [INPUT, MANDATORY, default=NO default value]

String database [INPUT, MANDATORY, default=NO default value]

Name of the Database to start from

String pName [INPUT, MANDATORY, default=NO default value]
PACS Level 0 Product Pool Name

String pType [INPUT, OPTIONAL, default=default value = "DBPool"]
Type of the generated Pool : "DBPool" or "LocalStore"

long hklimit [INPUT, OPTIONAL, default=default value = 10000]
Number of HK TmSourcePackets used for a Pool enty

long sclimit [INPUT, OPTIONAL, default=default value = 1000]
Number of Science TmSourcePackets used for a Pool enty

History

- 0.1 28-Mar-2007 EkW Initial version of this Task

1.157. pacsphoto_pipeline

Full Name:	herschel.pacs.spg.pipeline.oldstyle.pacsphoto.pacsphoto_pipeline
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle.pacsphoto import pacsphoto_pipeline

History

- 1.0 20080715 EkW initial version - basing on pacsphotpointsource.py
- 1.1 20080730 EkW add usage of PointinbProduct
- 1.2 20080821 EkW SPR 978 Usage of PointingProduct
- 1.3 20080911 EkW SPR 1037 Parallel Mode
- 2.0 20081023 EkW Pointing and FluxCalibration

1.158. pacsphotpointsource

Full Name:	herschel.pacs.spg.pipeline.oldstyle.pacsphotpointsource
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import pacsphotpointsource

History

- 1.0 20071219 EkW initial version
- 1.5 20080318 EkW major clean up
- 2.0 20080417 EkW re-write to match into pipeline architecture
- 2.2 20080830 EkW add PointingProduct
- 2.3 20080831 EkW move findBlocks
- 2.4 20081021 EkW Activate Flux Calibration / deglitching

1.159. pacsphotpointsourceIA

Full Name:	herschel.pacs.spg.pipeline.oldstyle.ipipe.pacsphotpointsourceIA
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle.ipipe import pacsphotpointsourceIA

History

- 1.0 20090219 EkW initial version from pacsphotpointsource.py
- 2.0 20091119 EkW adoption to current pipeline

1.160. pacsphotscanmap

Full Name:	herschel.pacs.spg.pipeline.oldstyle.pacsphotscanmap
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import pacsphotscanmap

History

- 1.0 20071219 EkW initial version
- 1.5 20080318 EkW major clean up
- 2.0 20080424 EkW re-write to match into pipeline architecture
- 2.1 20080715 EkW add usage of calTree
- 2.2 20080830 EkW add PointingProduct
- 2.3 20080831 EkW move findBlocks
- 2.4 20000309 EkW add filterSlew and filterOnTarget , remove cleanPlateau
- SPR 1337 optimizeOrientation = True

1.161. pacsphotscanmapIA

Full Name:	herschel.pacs.spg.pipeline.oldstyle.ipipe.pacsphotscanmapIA
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle.ipipe import pacsphotscanmapIA

History

- 1.0 20071219 EkW initial version
- 1.5 20080318 EkW major clean up
- 2.0 20080424 EkW re-write to match into pipeline architecture
- 2.1 20080715 EkW add usage of calTree
- 2.2 20080830 EkW add PointingProduct
- 2.3 20080831 EkW move findBlocks
- 2.4 20080904 Ekw For SVT use only scanmapsimple
- 3.0 20091119 EkW adopted to pipeline

1.162. pacsphotscanmapsimple

Full Name:	herschel.pacs.spg.pipeline.oldstyle.pacsphotscanmapsimple
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import pacsphotscanmapsimple

History

- 1.0 20071219 EkW initial version
- 1.5 20080318 EkW major clean up
- 2.0 20080424 EkW re-write to match into pipeline architecture
- 2.1 20080715 EkW add usage of calTree
- 2.2 20080830 EkW add PointingProduct
- 2.3 20080831 EkW move findBlocks
- 2.4 20080904 Ekw For SVT use only scanmapsimple
- 2.5 20000309 Ekw Add filter Slew and filterOnTarget, remove cleanPlateau
- 2.6 20090429 EkW calTree SPR 1552

1.163. pacsphotscanmapsimpleIA

Full Name:	herschel.pacs.spg.pipeline.oldstyle.ipipe.pacsphotscanmapsimpleIA
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle.ipipe import pacsphotscanmapsimpleIA

History

- 1.0 20071219 EkW initial version
- 1.5 20080318 EkW major clean up
- 2.0 20080424 EkW re-write to match into pipeline architecture
- 2.1 20080715 EkW add usage of calTree
- 2.2 20080830 EkW add PointingProduct
- 2.3 20080831 EkW move findBlocks
- 2.4 20080904 Ekw For SVT use only scanmapsimple
- 3.0 20091119 EkW adopted to pipeline

1.164. pacsphotsmallextended

Full Name:	herschel.pacs.spg.pipeline.oldstyle.pacsphotsmallextended
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import pacsphotsmallextended

History

- 1.0 20071219 EkW initial version
- 1.5 20080318 EkW major clean up
- 2.0 20080424 EkW re-write to match into pipeline architecture
- 2.1 20080830 EkW add PointingProduc
- 2.2 20080831 EkW move findBlocks
- 2.3 20080804 EkW SPR 960

1.165. pacsphotsmallextendedIA

Full Name:	herschel.pacs.spg.pipeline.oldstyle.ipipe.pacsphotsmallextendedIA
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle.ipipe import pacsphotsmallextendedIA

History

- 1.0 20090312 EkW initial version
- 2.0 20091119 EkW adoption to current pipeline

1.166. PacsQualityPlugin

Full Name:	herschel.pacs.spg.plugins.PacsQualityPlugin
Type:	Java Class -
Import:	from herschel.pacs.spg.plugins import PacsQualityPlugin

History

- 17 Mar 2008 EkW first draft
- 31 Jul 2008 JdJ added copying all quality flags from the frames to the quality context
- 22 Sep 2008 EkW SPR 1041, clean up and prepare STATE and ACTION
- 1.12 2008/12/04 ewieprec SPR 1213
- 1.11 2008/11/26 jdejong SPR 1094: fix for missing quality flags
- 1.10 2008/11/26 jdejong Changed description
- 1.9 2008/11/26 jdejong SPR 1094: changed the observation context keywords fore level-1 photometry
- 1.8 2008/11/12 jdejong branches: 1.8.2;
- 1.7 2008/11/11 jdejong adapted Quality Control handling to QC ICD version 1.1
- 1.6 2008/09/24 ewieprec PacsQualityPlugin.java
- 1.5 2008/07/31 jdejong Added copying quality flags to the quality context
- 1.4 2008/07/30 jdejong First implementation of QC population. Not tested.
- 1.3 2008/04/23 michael commented out unclear statements in PacsQualityPlugin to allow compilation
- 1.2 2008/04/16 ewieprec PacsQualityPlugin.java
- 1.1 2008/04/16 ewieprec first draft
- 1.8.2.2 2008/12/05 ewieprec SPR 1213
- 1.8.2.1 2008/12/03 ewieprec SPR 1094

1.167. PacsRebinnedCube

Full Name:	herschel.pacs.signal.PacsRebinnedCube
Type:	Java Class -
Import:	from herschel.pacs.signal import PacsRebinnedCube

Description

TBW

Example

Example 1: Untitled

```
Looking at a rebinned cube.
```

API Summary

Properties
PacsCube **pacsCube** [INPUT, MANDATORY, default=NO default value]
PacsRebinnedCube **PacsRebinnedCube** [OUTPUT, MANDATORY, default=NO default value]

API details

Properties

PacsCube pacsCube [INPUT, MANDATORY, default=NO default value]

input PacsCube object

PacsRebinnedCube PacsRebinnedCube [OUTPUT, MANDATORY, default=NO default value]

returned cube with mxnx5x5 cube representing the browse quality cube.

1.168. PacsSliceContextTask

Full Name:	herschel.pacs.spg.PacsSliceContextTask
Type:	Java Task -
Import:	from herschel.pacs.spg import PacsSliceContextTask

Description

(Re)slices a given context into a new context.

This task (re)slices a given context into a new context using rules on the blocktables to select the appropriate blocks for each slice.

A rule indicates which column in the blocktable has to change and after how many changes a new Frames object is started.

API Summary

Properties
`ListContext` **`inListContext`** `[INPUT, MANDATORY, default=no default]`
`Object[]` **`slicingRules`** `[INPUT, OPTIONAL, default=no default]`

API details

Properties

`ListContext inListContext [INPUT, MANDATORY, default=no default]`

List context which needs to be (re)sliced

`Object[] slicingRules [INPUT, OPTIONAL, default=no default]`

A list of slicing rules. Each rule consists of a list with the column name and the number of allowed changes in this column. The format is ["ColName",noChanges]. So, a complete parameter value would be e.g. [["NoddingPosition",1]]

1.169. pacsspeccalblock

Full Name:	herschel.pacs.spg.pipeline.oldstyle.pacsspeccalblock
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import pacsspeccalblock

History

- 1.0 20090312 JS initial version

1.170. pacsspecchopnodmap

Full Name:	herschel.pacs.spg.pipeline.oldstyle.pacsspecchopnodmap
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import pacsspecchopnodmap

History

- 1.0 20071019 JS initial version
- 1.1 20080222 JS adapted according to new cal framework
- 1.2 20080311 JS adapt sequence of modules and add specAssignRaDec
- 2.0 20080430 JS re-write to match into pipeline architecture
- 2.1 20080721 JS split into 4 spectrometer pipelines according to AOR differences

1.171. pacsspecchopnodstarframesIA

Full Name:	herschel.pacs.spg.pipeline.oldstyle.ipipe.pacsspecchopnodstarframesIA
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle.ipipe import pacsspecchopnodstarframesIA

History

- 1.0 20090313 JS initial version
- 1.1 20090624 JS updated

1.172. pacsspecchopnodstar

Full Name:	herschel.pacs.spg.pipeline.oldstyle.pacsspecchopnodstar
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import pacsspecchopnodstar

History

- 1.0 20071019 JS initial version
- 1.1 20080222 JS adapted according to new cal framework
- 1.2 20080311 JS adapt sequence of modules and add specAssignRaDec
- 2.0 20080430 JS re-write to match into pipeline architecture
- 2.1 20080721 JS split into 4 spectrometer pipelines according to AOR differences

1.173. pacsspecchopnodstarrampsIA

Full Name:	herschel.pacs.spg.pipeline.oldstyle.ipipe.pacsspecchopnodstarrampsIA
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle.ipipe import pacsspecchopnodstarrampsIA

History

- 1.0 20090313 JS initial version
- 1.1 20090624 JS updated

1.174. pacsspecframes

Full Name:	herschel.pacs.spg.pipeline.oldstyle.pacsspecframes
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import pacsspecframes

History

- 1.0 20071019 JS initial version
- 1.1 20080222 JS adapted according to new cal framework
- 1.2 20080311 JS adapt sequence of modules and add specAssignRaDec
- 2.0 20080430 JS re-write to match into pipeline architecture
- 2.1 20080730 JS add PointingProduct

1.175. pacsspecoffmap

Full Name:	herschel.pacs.spg.pipeline.oldstyle.pacsspecoffmap
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import pacsspecoffmap

History

- 1.0 20071019 JS initial version
- 1.1 20080222 JS adapted according to new cal framework
- 1.2 20080311 JS adapt sequence of modules and add specAssignRaDec
- 2.0 20080430 JS re-write to match into pipeline architecture
- 2.1 20080721 JS split into 4 spectrometer pipelines according to AOR differences

1.176. pacsspecoffmapIA

Full Name:	herschel.pacs.spg.pipeline.oldstyle.ipipe.pacsspecoffmapIA
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle.ipipe import pacsspecoffmapIA

History

- 1.0 20090318 JS initial version
- 1.1 20090624 JS updated

1.177. pacsspecramps

Full Name:	herschel.pacs.spg.pipeline.oldstyle.pacsspecramps
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import pacsspecramps

History

- 1.0 20071019 JS initial version
- 1.1 20080222 JS adapted according to new cal framework
- 1.2 20080311 JS adapt sequence of modules and add specAssignRaDec
- 2.0 20080430 JS re-write to match into pipeline architecture

1.178. pacsspectro_pipeline

Full Name:	herschel.pacs.spg.pipeline.oldstyle.pacsspectro.pacsspectro_pipeline
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle.pacsspectro import pacsspectro_pipeline

History

- 1.0 20080430 JS initial version
- 2.0 20080605 JS combine ramps and frames processing in one file
- 2.1 20080721 JS split into 4 spectrometer pipelines according to AOR differences
- 2.2 20080730 EkW Clean up / SPR 866
- 2.3 20080828 JS insert extract pointing product
- 2.4 20081202 JS implementation of SPR 1197
- 2.5 20081209 JS implementation of SPR 1225
- 2.6 20090311 JS implementation of slicing and cleaning
- 2.7 20090605 JS+JdeJ implementation of SPR 1665

1.179. pacsspecwaveswitch

Full Name:	herschel.pacs.spg.pipeline.oldstyle.pacsspecwaveswitch
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import pacsspecwaveswitch

History

- 1.0 20071019 JS initial version
- 1.1 20080222 JS adapted according to new cal framework
- 1.2 20080311 JS adapt sequence of modules and add specAssignRaDec
- 2.0 20080430 JS re-write to match into pipeline architecture
- 2.1 20080721 JS split into 4 spectrometer pipelines according to AOR differences

1.180. pacsspecwaveswitchIA

Full Name:	herschel.pacs.spg.pipeline.oldstyle.ipipe.pacsspecwaveswitchIA
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle.ipipe import pacsspecwaveswitchIA

History

- 1.0 20090318 JS initial version
- 1.1 20090624 JS updated

1.181. PacsTodProduct

Full Name:	herschel.pacs.signal.PacsTodProduct
Type:	Java Class -
Import:	from herschel.pacs.signal import PacsTodProduct

Description

PacsTodProduct

Product that stores tod pertinent information. Tod is used in Madmap making.

API Summary

Constructor
PacsTodProduct() PacsTodProduct

Methods
setTodFile(String filename) setTodFile
String **getTodFile**() getTodFile
setUniqueNumSkyPixels(int npix) setUniqueNumSkyPixels
int **getUniqueNumSkyPixels**() getUniqueNumSkyPixels
setUniqueSkyPixelIndex(Int1d uniqueSkyPixelIndex) setUniqueSkyPixelIndex
Int1d **getUniqueSkyPixelIndex**() getUniqueSkyPixelIndex
setTodFromIndex(Long2d from) setTodFromIndex
Long2d **getTodFromIndex**() getTodFromIndex
setTodToIndex(Long2d to) setTodToIndex
Long2d **getTodToIndex**() getTodToIndex
setSkyMap(Double1d map) setSkyMap
Double1d **getSkyMap**() getSkyMap
setWcs(Header hdr) setWcs

Methods
`Wcs` **`getWcs`**`()` getWcs
`Header` **`getHeader`**`()` getHeader
`String` **`getBand`**`()` getBand
`setBand``(String band)` setBand
`Boolean` **`getMode`**`()` getMode
`setMode``(boolean mode)` setMode
`cpTodArray``()` cpTodArray

API details

Constructor

`PacsTodProduct()`
PacsTodProduct Default constructor

Methods

`setTodFile(String filename)`
setTodFile Set filename for output tod file **Argument** `String` **`filename`** `[INPUT, MANDATORY, default=no default value]` Tod filename

`String getTodFile()`
getTodFile Get tod filename **Return** **`String`** Tod filename **Errors** `NoSuchFieldException` When tod filename can't be found in the product

setUniqueNumSkyPixels(int npix)

setUniqueNumSkyPixels

Set total number of unique sky pixels

Argument

`int` **`npix`** `[INPUT, MANDATORY, default=no default value]`

Total number of unique sky pixels

int getUniqueNumSkyPixels()

getUniqueNumSkyPixels

Returns total number of unique sky pixels

Return

int

Total number of unique sky pixels

Errors

`NoSuchFieldException`

When NumSkyPixels is not found

setUniqueSkyPixelIndex(Int1d uniqueSkyPixelIndex)

setUniqueSkyPixelIndex

Set the unique sky pixel index

Argument

`Int1d` **`uniqueSkyPixelIndex`** `[INPUT, MANDATORY, default=no default value]`

Unique sky pixel index

Int1d getUniqueSkyPixelIndex()

getUniqueSkyPixelIndex

Get the unique sky pixel index

Return

Int1d

The unique sky pixel index

Errors

`NoSuchFieldException`

When UniqueSkyPixelIndex is not found

setTodFromIndex(Long2d from)

setTodFromIndex

Set Tod's from index

Argument

`Long2d` **`from`** `[INPUT, MANDATORY, default=no default value]`

setTodFromIndex(Long2d from)

Tod's from index

Long2d getTodFromIndex()

getTodFromIndex

Get tod's from index

Return

Long2d

Tod's from index

Errors

`NoSuchFieldException`

When FromIndex is not found

setTodToIndex(Long2d to)

setTodToIndex

Set Tod's to index

Argument

`Long2d` **`to`** `[INPUT, MANDATORY, default=no default value]`

Tod's to index

Long2d getTodToIndex()

getTodToIndex

Get tod's to index

Return

Long2d

Tod's to index

Errors

`NoSuchFieldException`

When ToIndex is not found

setSkyMap(Double1d map)

setSkyMap

Set the map data returned from MadMapper

Argument

`Double1d` **`map`** `[INPUT, MANDATORY, default=no default value]`

Map data returned from MadMapper

Double1d getSkyMap()

getSkyMap

Get the map data returned from MadMapper

`Double1d getSkyMap()`

Return

Double1d

The map data

Errors

`NoSuchFieldException`

When SkyMap is not found

`setWcs(Header hdr)`

setWcs

Set Wcs based Fits Header information

Argument

`Header` **`hdr`** `[INPUT, MANDATORY, default=no default value]`

nom.tam.fits.Header type header

`Wcs getWcs()`

getWcs

Return Wcs

Return

Wcs

Wcs

`Header getHeader()`

getHeader

Return nom.tam.fits.Header type header

Return

Header

nom.tam.fits.Header type header

Errors

`HeaderCardException`

When failed at adding new card to header

`String getBand()`

getBand

Get BAND value

Return

String

BAND value

Errors

String getBand()

NoSuchFieldException

When BAND is not found

setBand(String band)

setBand

Set BAND value, such as 'BL', "BS', and "R"

Argument

String **band** [INPUT, MANDATORY, default=no default value]

BAND value

Boolean getMode()

getMode

Return true for SpirePacsParallel mode, false otherwise

Return

Boolean

True for SpirePacsParallel mode, false otherwise

Errors

NoSuchFieldException

When MODE is not found

setMode(boolean mode)

setMode

Set MODE value, true for SpirePacsParallel mode, false otherwise

Argument

boolean **mode** [INPUT, MANDATORY, default=no default value]

MODE value

cpTodArray()

cpTodArray

Copy Tod array from the temp file to one with .save extension

History

- 2008-02-25 - CL: Make all data as Datasets and Metadata, v1.12
- 2008-02-26 - CL: Add copy() method, add two more Datasets, v1.13
- 2008-03-04 - CL: Add Wcs get method, v1.14
- 2008-05-12 - CL: Add functions to get/set wavelength band, v1.15
- 2008-06-12 - CL: Fix SPR 4367- remove dependency on herschel.ia.image.ImageDataset, v1.16

- `2008-07-24 - CL`: Add new metadata, v1.17
- `2008-10-31 - CL`: Replace Tod to and from index with 2 Long2d arrays, v1.18

1.182. PairDiffHodLehEstTask

Full Name:	herschel.pacs.toolboxes.spec.PairDiffHodLehEstTask
Type:	Java Task -
Import:	from herschel.pacs.toolboxes.spec import PairDiffHodLehEstTask

Description

function that converts a Ramps object containing ramps or averaged

ramps to a Frames object after calculating the Hodges-Lehmann estimator of the pairwise differences. This function computes the pairwise differences of each ramp and then calculates the Hodges-Lehmann estimator to get the mean signal. The H.-L. estimator is defined as the median of the mean of pairs of a pairwise differences array. Masks are applied as follows: a master mask is constructed using the already available masks in the Ramps object. The master mask is then applied to the indices and readouts before the signal is calculated. As x-axis the fine time information per sample is used. This works also for averaged ramps (sub-ramps). The units of the output signals is corresponding to the onboard fitting mode readouts per reset interval. The status parameters are also populated using the first elements of the ramp status arrays The stdev array is populated using the standard deviation of the array of pairwise means of pairwise differences.

API Summary

Jython Syntax
`Frames outFrame = pairDiffHodLehEst(Ramps inRamp)`

Property
`Ramps inRamp [INPUT, MANDATORY, default=NO default value]`

API details

Property

`Ramps inRamp [INPUT, MANDATORY, default=NO default value]`
input Ramps object

History

- 1.0 20050517 JS initial version
- 1.1 20050524 JS fine time replaced by crcrmp values, imports shifted inside function
- 1.2 20050620 JS mask handling introduced, crcrmp values replaced by indices
- 1.3 20050621 JS bug fix
- 1.4 20050805 JS header adopted to jtags syntax
- 1.5 20051107 JS clean up of imports
- 1.6 20051214 JS conversion to task, populating status, improved time handling
- 1.7 20060306 JS getRampValues() -> getSignal()

- 1.8 20060316 JS stdev of mean signal is set in Frames
- camera parameter is read directly from Ramps, not input parameter anymore
- 1.9 20060327 JS status parameter conversion distinguishes 1d and 2d arrays
- 1.10 20060518 JS bug fix
- 1.11 20070116 JS improved version and converted to java
- 1.12 20080519 JS implementation of SPR 817, additional optional parameter
- 1.13 20080611 JS implementation of SPR 887, optional boolean parameters ignore...Mask added
- 1.14 20081010 JS implementation of SPR 1080
- Calculating the averaged pairwise differences of Ramps using the Hodges-Lehmann estimator
- and converting to Frames

1.183. PairDiffSigClipTask

Full Name:	herschel.pacs.toolboxes.spec.PairDiffSigClipTask
Type:	Java Task -
Import:	from herschel.pacs.toolboxes.spec import PairDiffSigClipTask

Description

function that converts a Ramps object containing ramps or averaged

ramps to a Frames object after calculating the averaged pairwise differences. This function computes the pairwise differences of each ramp and then calculates the mean by discarding a factor of sigma of the low and high difference values. Masks are applied as follows: a master mask is constructed using the already available masks in the Ramps object. The master mask is then applied to the indices and readouts before the signal is calculated. As x-axis the fine time information per sample is used. This works also for averaged ramps (sub-ramps). The units of the output signals is corresponding to the onboard fitting mode readouts per reset interval. The status parameters are also populated using the first elements of the ramp status arrays The stdev array is populated using the standard deviation of the sigma clipped array

API Summary

Jython Syntax
`Frames outFrame = pairDiffSigClip(Ramps inRamp [, Double sigma = <sigma>]))`

Properties
`Ramps inRamp [INPUT, MANDATORY, default=NO default value]`
`Double sigma [INPUT, OPTIONAL, default=3.]`

API details

Properties

`Ramps inRamp [INPUT, MANDATORY, default=NO default value]`
input Ramps object

`Double sigma [INPUT, OPTIONAL, default=3.]`
optional: sigma factor for rejected values to average, default: 3.

History

- 1.0 20050517 JS initial version
- 1.1 20050524 JS fine time replaced by crcrmp values, imports shifted inside function
- 1.2 20050620 JS mask handling introduced, crcrmp values replaced by indices
- 1.3 20050621 JS bug fix
- 1.4 20050805 JS header adopted to jtags syntax

- 1.5 20051107 JS clean up of imports
- 1.6 20051214 JS conversion to task, populating status, improved time handling
- 1.7 20060306 JS getRampValues() -> getSignal()
- 1.8 20060316 JS stdev of mean signal is set in Frames
- camera parameter is read directly from Ramps, not input parameter anymore
- 1.9 20060327 JS status parameter conversion distinguishes 1d and 2d arrays
- 1.10 20060518 JS bug fix
- 1.11 20070116 JS improved version and converted to java
- 1.12 20080519 JS implementation of SPR 817, additional optional parameter
- 1.13 20080611 JS implementation of SPR 887, optional boolean parameters ignore...Mask added
- 1.14 20081010 JS implementation of SPR 1080
- Calculating the averaged pairwise differences of Ramps and converting to Frames

1.184. percClipMean

Full Name:	herschel.pacs.toolboxes.numeric.percClipMean
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import percClipMean

Description

numeric function that calculates a percentile clipped mean of a double array

This function first sorts the input array values. Afterwards, it averages the array after discarding a given percentage of the highest values defined by the input parameter hiReject and a given percentage of the lowest values defined by the input parameter loReject. Remark: loReject + hiReject must be smaller than 100.

API Summary

Jython Syntax

```
Double average = percClipMean(Double1d array, Double loReject,
Double hiReject)
```

Properties

Double1d **array** [INPUT, MANDATORY, default=NO default value]

Double **loReject** [INPUT, MANDATORY, default=NO default value]

Double **hiReject** [INPUT, MANDATORY, default=NO default value]

Double **average** [OUTPUT, MANDATORY, default=NO default value]

API details

Properties

Double1d array [INPUT, MANDATORY, default=NO default value]

input Double1d array

Double loReject [INPUT, MANDATORY, default=NO default value]

percentage of lowest values to discard

Double hiReject [INPUT, MANDATORY, default=NO default value]

percentage of highest values to discard

Double average [OUTPUT, MANDATORY, default=NO default value]

returned average value

History

- 1.0 21-Apr-2005 JS initial version
- 1.1 20050524 JS imports moved inside function
- 1.2 20050707 JS necessary numeric imports added

- 1.3 20050803 JS header adopted to jtags concept
- 1.4 20050829 JS toolbox.basic imported
- 1.5 20070123 JS performance improved version

1.185. percClipMedian

Full Name:	herschel.pacs.toolboxes.numeric.percClipMedian
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import percClipMedian

Description

numeric function that calculates a percentile clipped median of a double array

This function first sorts the input array values. Afterwards, it calculates the median after discarding a given percentage of the highest values defined by the input parameter hiReject and a given percentage of the lowest values defined by the input parameter loReject. Remark: loReject + hiReject must be smaller than 100.

API Summary

Jython Syntax

```
Double med = percClipMedian(Double1d array, Double loReject,
Double hiReject)
```

Properties

Double1d **array** [INPUT, MANDATORY, default=NO default value]

Double **loReject** [INPUT, MANDATORY, default=NO default value]

Double **hiReject** [INPUT, MANDATORY, default=NO default value]

Double **med** [OUTPUT, MANDATORY, default=NO default value]

API details

Properties

Double1d array [INPUT, MANDATORY, default=NO default value]

input Double1d array

Double loReject [INPUT, MANDATORY, default=NO default value]

percentage of lowest values to discard

Double hiReject [INPUT, MANDATORY, default=NO default value]

percentage of highest values to discard

Double med [OUTPUT, MANDATORY, default=NO default value]

returned median value

History

- 1.0 21-Apr-2005 JS initial version
- 1.1 20050524 JS imports moved inside function
- 1.2 20050629 JS bug fix

- 1.3 20050707 JS necessary numeric imports added
- 1.4 20050803 JS header adopted to jtags concept
- 1.5 20050829 JS toolbox.basic imported
- 1.6 20070123 JS performance improved version

1.186. percClipStdev

Full Name:	herschel.pacs.toolboxes.numeric.percClipStdev
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import percClipStdev

Description

numeric function that calculates a percentile clipped standard deviation of a double array

This function first sorts the input array values. Afterwards, it calculates the standard deviation after discarding a given percentage of the highest values defined by the input parameter hiReject and a given percentage of the lowest values defined by the input parameter loReject. Remark: loReject + hiReject must be smaller than 100.

API Summary

Jython Syntax
`Double sig = percClipStdev(Double1d array, Double loReject, Double hiReject)`

Properties
`Double1d array [INPUT, MANDATORY, default=NO default value]`
`Double loReject [INPUT, MANDATORY, default=NO default value]`
`Double hiReject [INPUT, MANDATORY, default=NO default value]`
`Double sig [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

Double1d array [INPUT, MANDATORY, default=NO default value]
input Double1d array

Double loReject [INPUT, MANDATORY, default=NO default value]
percentage of lowest values to discard

Double hiReject [INPUT, MANDATORY, default=NO default value]
percentage of highest values to discard

Double sig [OUTPUT, MANDATORY, default=NO default value]
returned standard deviation value

History

- 1.0 07-Juli-2005 JS initial version
- 1.1 04-Aug-2005 JS header adopted to jtags concept
- 1.2 29-Aug-2005 JS toolbox.basic imported

- 1.3 23-Jan-2007 JS performance improved version

1.187. PhotAddInstantPointingTask

Full Name:	herschel.pacs.spg.phot.PhotAddInstantPointingTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.phot import PhotAddInstantPointingTask

Description

Include s/c pointing as coordinates and additonal pointing information like scanNumber in PACS frames product.

By default the Filtered Pointing information is used, but also the gyro propagated pointing information may be used.

This is done by using the Frames status entry FINETIME and extract the associated information from the PointingProduct.

Also the SIAM matrix is applied and aberration is done (if the proper Products are passed)

The result is added to the status entry of the Frames Product

API Summary

Jython Syntax
`Frames outFrames = photAddInstantPointing(inFrames, pp [calTree] [siam] [orbitEphem] [horizons] [isSso] [noInter] [useGyro] [copy])`

Properties
`Frames inFrames [INPUT, MANDATORY, default=NO default]`
`PointingProduct pp [INPUT, MANDATORY, default=NO]`
`PacsCal calTree [INPUT, OPTIONAL, default=None]`
`Siam siam [INPUT, OPTIONAL, default=None]`
`OrbitEphemerisProduct orbitEpehm [INPUT, OPTIONAL, default=None]`
`Horizons horizons [INPUT, OPTIONAL, default=None]`
`Boolean isSso [INPUT, OPTIONAL, default=False]`
`Boolean useGyro [INPUT, OPTIONAL, default=False]`
`Integer copy [INPUT, OPTIONAL, default=None]`
`Frames outFrames [OUTPUT, MANDATORY, default=NO default value Output Frames]`

API details

Properties

`Frames inFrames [INPUT, MANDATORY, default=NO default]`
value Input Frames class

`PointingProduct pp [INPUT, MANDATORY, default=NO]`
default value Pointing Product

PacsCal calTree [INPUT, OPTIONAL, default=None]

Pacs Calibration Tree

Siam siam [INPUT, OPTIONAL, default=None]

SIAM Product

OrbitEphemerisProduct orbitEpehm [INPUT, OPTIONAL, default=None]

Orbit Ephemeris Product needed for aberration correction of non SSO objects

Horizons horizons [INPUT, OPTIONAL, default=None]

Horizons Product (Needed for aberration correction of SSO objects)

Boolean isSso [INPUT, OPTIONAL, default=False]

Is it a solar system Object ?

Boolean useGyro [INPUT, OPTIONAL, default=False]

Use the Gyro propagated information instead of the filtered

Integer copy [INPUT, OPTIONAL, default=None]

Copy the Frames class instead of just adding the status word

Frames outFrames [OUTPUT, MANDATORY, default=NO default value Output Frames]

Output frames (default it is the reference of input Frames + new status entries)

1.188. PhotAddPointings4PointSourceTask

Full Name:	herschel.pacs.spg.phot.PhotAddPointings4PointSourceTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.phot import PhotAddPointings4PointSourceTask

Description

Extract Pointing information for further Photometer PointSource processing

Store the averaged ra,dec of the virtual aperture for both nod positions, dither positions and chop positions

Add the PhotPointSource Dataset to the Frames class. It contains per nod position, dither position and chopper position the first value of : RaArray, DecArray, PaArray, CPR, DithPos, NodCycleNum, ChopperPlateau, isAPosition

This information is later used on PhotProjectPointSource to map the Frames.

API Summary

Jython Syntax
`frames = photAddPointings4PointSource(frames)`

Property
`Frames frames [IN/OUT, MANDATORY, default=NO default value]`

API details

Property

`Frames frames [IN/OUT, MANDATORY, default=NO default value]`
Frames containing detector data are converted from digits to volts

1.189. PhotAssignRaDecTask

Full Name:	herschel.pacs.spg.phot.PhotAssignRaDecTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import PhotAssignRaDecTask

Description

Assign RA and Dec position for every individul pixel (currently working with XY coordinates)

API Summary

Jython Syntax
`Frames outFrame = photAssignRaDec(Frames inFrame [,calTree=calTree] [arrayInstrument=arrayInstrument] [,subarrayArray=subarrayArray] [,copy=copy] )`

Properties
`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
`Integer copy [INPUT, OPTIONAL, default=0]`

API details

Properties

`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
input Frames object

`Integer copy [INPUT, OPTIONAL, default=0]`
copy : 0 (default) - give back a reference, 1: give back a copy

1.190. PhotConvDigit2VoltsTask

Full Name:	herschel.pacs.spg.phot.PhotConvDigit2VoltsTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.phot import PhotConvDigit2VoltsTask
Category:	pacs/spg/phot

Description

Convert Digits to Volts

Example

Example 1: flag saturation

```
frames = photConvDigit2Volts(frames)
```

API Summary

Jython Syntax

```
frames = photConvDigit2Volts(frames,calTree=calTree,
photGain=photGain, copy =copy)
```

Properties

Frames **frames** [IN/OUT, MANDATORY, default=NO default value]

String **calTree** [INPUT, OPTIONAL, default=default value null]

String **photGain** [INPUT, OPTIONAL, default=default value null]

String **copy** [INPUT, OPTIONAL, default=default value : 0]

Miscellaneous

Gain extraction shall be done on Frames generation level

API details

Properties

Frames frames [IN/OUT, MANDATORY, default=NO default value]

Frames containing detector data are converted from digits to volts

String calTree [INPUT, OPTIONAL, default=default value null]

Calibration Tree

String photGain [INPUT, OPTIONAL, default=default value null]

PhotGain Calibration data, calTree entries are overwritten)

String copy [INPUT, OPTIONAL, default=default value : 0]

Copy the input (1) and generate new output or overwrite the input (0)

History

- 02-Nov-2005 EkW Initial version of this Task, basing on Koryos prototype
- 07-Nov-2005 JS clean up of imports
- 09-Nov-2005 EkW - Use calibration files
- 08-Dec-2005 EkW - Include comments from KO - add copy keyword
- 20-Nov-2005 EkW - documentation
- 12-Dec-2005 EkW - include Koryo comment
- 21-Jul-2005 EkW - SPR 555 and 557
- 07-Aug-2006 BM - Add differential mode, gain, offset - FM detectors
- 30-Aug-2006 JS - Task mode=... replaced by type IN/OUT/IO
- 14-Sep-2006 BM minor changes
- `2006-11-27 - BM`: enhanced version - modes not supported raise an exception - and SPR 631
- 01-May-2007 EkW
- 28-Jan-2008 BM - added unit
- 21-Feb-2008 EkW - adopt to new calibration framework / documentation
- 28-Feb-2008 BM - change outFrames.getStartTime() into outFrames.getStartDate()

1.191. PhotCorrectCrosstalkTask

Full Name:	herschel.pacs.spg.phot.PhotCorrectCrosstalkTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import PhotCorrectCrosstalkTask

Description

JAVA Task as it need loops

This task reads crosstalk ratios from a calibration file, uses it to calculate crosstalk and subtracts from every pixel the signal fraction of a crosstalking pixel.

The structure of the calibration file

The calibration file should be created and edited with the script makePhotCalCrosstalkMatrix.py.

The Calfile contains an ArrayDataset with two Double2d arrays in it, one for the red and one for the blue channel.

Dimensions of the Double2d are [number of cross-talks found, 5).

The values are composed like this:

- [k,0] : the row value of the original pixel
- [k,1] : the column value of the original pixel
- [k,2] : the row value of the crosstalking pixel
- [k,3] : the column value of the crosstalking pixel
- [k,4] : the crosstalking ratio

Example

lets assume the signal of pixel (4,7) appears with 50% in pixel (0,0) and the signal of pixel (8, 10) appears with 13% in pixel (5,1). Then the array looks like this:

[0,0] = 4

[0,1] = 7

[0,2] = 0

[0,3] = 0

[0,4] = 0.5

[1,0] = 8

[1,1] = 10

[1,2] = 5

[1,3] = 1

[1,4] = 0.13

Thus the crosstalk corrected values of pixel (0,0) and (5,1) would be

realsig(0,0) = realsig(0,0) - 0.5*sig(4,7)

realsig(5,1) = realsig(5,1) - 0.13*sig(8,10)

These simple subtractions are applied by this task for the whole signal array.

API Summary

Jython Syntax
`Frames outFrame = specCorrectCrosstalk(Frames inFrame [, PacsCal calTree]`
`[,CrosstalkMatrix crosstalkMatrix][, int copy][, QualityContext qualityContext])`

Properties
`Frames` **`inFrame`** `[INPUT, MANDATORY, default=NO default value]`
`PacsCal` **`calTree`** `[INPUT, Optional, default=default value: None]`
`CrosstalkMatrix` **`crosstalkMatrix`** `[INPUT, Optional, default=default value: None]`
`Integer` **`copy`** `[INPUT, Optional, default=default value: 0]`
`QualityContext` **`qualityContext`** `[INOUT, Optional, default=default value: None]`
`Frames` **`outFrame`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
input Frames object

`PacsCal calTree [INPUT, Optional, default=default value: None]`
a PacsCal object for calfile access

`CrosstalkMatrix crosstalkMatrix [INPUT, Optional, default=default value: None]`
cross talk matrix the name of the calibration file that contains the crosstalk data

`Integer copy [INPUT, Optional, default=default value: 0]`
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

`QualityContext qualityContext [INOUT, Optional, default=default value: None]`
quality control flag

`Frames outFrame [OUTPUT, MANDATORY, default=NO default value]`
returned Frames object containing the cross talk corrected signals

1.192. PhotCorrZeroLevelTask

Full Name:	herschel.pacs.toolboxes.phot.PhotCorrZeroLevelTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.phot import PhotCorrZeroLevelTask

Description

Returns Frames whose Signal is subtracted by zero level calibration values

API Summary

Jython Syntax
`Frames frame = photCorrZeroLevel(Frames inFrames [, copy=copy] [, calTree=calTree] [,corrZeroLevel=corrZeroLevel])`

Properties
`Frames` **`inFrames`** `[INPUT, MANDATORY, default=NO default value]`
`Integer` **`copy`** `[INPUT, Optional, default=default value: 0]`
`CalibrationTree` **`calTree`** `[INPUT, Optional, default=NO default value]`
`CorrZeroLevel` **`corrZeroLevel`** `[INPUT, Optional, default=NO default value]`
`Frames` **`outFrame`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Frames inFrames [INPUT, MANDATORY, default=NO default value]`
input Frames object with signal, whose dimensions are (y,x,z), or (rows, columns, frames)

`Integer copy [INPUT, Optional, default=default value: 0]`
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

`CalibrationTree calTree [INPUT, Optional, default=NO default value]`
a pre-specified CalibrationTree

`CorrZeroLevel corrZeroLevel [INPUT, Optional, default=NO default value]`
a pre-specified CorrZeroLevel

`Frames outFrame [OUTPUT, MANDATORY, default=NO default value]`
returned frame with calibration values subtracted

History

- 1.0 14-Jan-2007 JJ initial version
- 1.1 22-Jan-2007 Fix documentation, add import java String
- 1.2 23-Jan-2007 Signal dimensions are row, column, frames
- 1.3 23-Jan-2007 Calibration files for both BLUE and RED contain negative
- numbers that need to be added, not subtracted.
- 1.3 02-Dec-2008 Update to use current calibration framework, SCR-1170

1.193. PhotCSProcessingTask

Full Name:	herschel.pacs.spg.phot.PhotCSProcessingTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.phot import PhotCSProcessingTask
Category:	pacs/spg/phot

Description

Jython task process calibration blocks found in the telemetry

Example

Example 1: from spg import *

```
photCSProcessing(inTrendProducts,hkdata)
```

API Summary

Jython Syntax

```
outTrendProducts =
photCSProcessing(inTrendProducts,hkdata,caltree[sigclip=True|
False][,nsgima=n][,quality])
```

Properties

type **inTrendProducts is a PhotTrendProducts object returned by PhotCSExtraction task** [INPUT, MANDATORY, default=no default value]

type **hkdata is a TableDataset** [INPUT, MANDATORY, default=no default value]

Limitations

No Limitation

API details

Properties

type inTrendProducts is a PhotTrendProducts object returned by PhotCSExtraction task [INPUT, MANDATORY, default=no default value]

type hkdata is a TableDataset [INPUT, MANDATORY, default=no default value]

See also

- ???

History

- 20081114 BM initial version

- 20081117 BM first running version with PhotCSDiffTask
- 20081120 BM replace obsContext by MapContext containing frames
- 20080121 BM adapted to work with PhotCSProducts class
- 20090204 BM adapted to work with PhotTrendProducts class
- 20090210 BM added calTree as parameter
- 20091106 BM renamed PhotDiffCalTask to PhotCSDiffTask

1.194. PhotCSProducts

Full Name:	herschel.pacs.signal.PhotCSProducts
Type:	Java Class -
Import:	from herschel.pacs.signal import PhotCSProducts

History

- 2009-01-22 : (bm) initial version

1.195. PhotDiffCStoringTask

Full Name:	herschel.pacs.toolboxes.phot.PhotDiffCStoringTask
Type:	Java Task -
Import:	from herschel.pacs.toolboxes.phot import PhotDiffCStoringTask
Category:	class

API Summary

Jython Syntax
`outFrames = PhotDiffCStoringTask(inFrames [,newVersion=newVersion] [,quality=quality][,copy=copy])`

Properties
`type` **`inFrames : the input frame object containing the difference and the average of the images per plateau.`** `[INPUT, MANDATORY, default=no default value]`
`type` **`newVersion: forces the creation of a new dCS version in the pool storage`** `[INPUT, MANDATORY, default=no default value]`
`type` **`copy : boolean with the possible values :`** `[INPUT, MANDATORY, default=no default value]`
`type` **`quality : the quality context`** `[INPUT, MANDATORY, default=no default value]`

API details

Properties

`type inFrames : the input frame object containing the difference and the average of the images per plateau. [INPUT, MANDATORY, default=no default value]`

`type newVersion: forces the creation of a new dCS version in the pool storage [INPUT, MANDATORY, default=no default value]`

* false (jython: 0) - no new version is created * true (jython: 1) - new version is created

`type copy : boolean with the possible values : [INPUT, MANDATORY, default=no default value]`

* false (jython: 0) - inFrames will be modified and returned * true (jython: 1) - a copy of inFrames will be returned

`type quality : the quality context [INPUT, MANDATORY, default=no default value]`

History

- `2008-04-18 - BM`: : initial version
- `2008-27-24 - BM`: : change the keyword name channel into Channel

- $Log: PhotDiffCStoringTask.java,v $
- Revision 1.2 2010/02/18 13:13:54 jdejong
- PACS-2369: pacs_spg import updates after moving code
- Revision 1.1 2010/02/11 10:36:14 wim
- Code reorganisation
- Revision 1.8 2010/02/02 11:34:21 hsclib
- LGPL
- Revision 1.7 2010/01/28 16:38:43 juergen
- SPR 999
- in case of error: throws RuntimeException
- Revision 1.6 2008/12/10 16:39:03 bmorin
- added version

1.196. PhotDriftCorrectionTask

Full Name:	herschel.pacs.spg.phot.PhotDriftCorrectionTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import PhotDriftCorrectionTask
Category:	class

API Summary

Jython Syntax
`outFrames = PhotDriftCorrection(inFrames [,algo=algo] [,quality=quality][,copy=copy])`

Properties
type **inFrames : the input frame object containing stack of images at** [INPUT, MANDATORY, default=no default value]
type **algo : algorithm used to compute dCS when the frames contains several calibration blocks** [possible values :, MANDATORY, default=no default value]
type **dCSRef : Difference of the calibration source got during flat field computation** [INPUT, MANDATORY, default=no default value]
type **threshold : threshold used to trigger an alert when dCsRef/dCS ratio is too big or too small** [INPUT, MANDATORY, default=no default value]
type **copy : boolean with the possible values :** [INPUT, MANDATORY, default=no default value]
type **quality : the quality context** [INPUT, MANDATORY, default=no default value]

API details

Properties

type inFrames : the input frame object containing stack of images at [INPUT, MANDATORY, default=no default value]

dither position. All ON-OFF chopping data has been identified, substracted each other and averaged.

type algo : algorithm used to compute dCS when the frames contains several calibration blocks [possible values :, MANDATORY, default=no default value]

* FIRST [default]- first calibration block is used * MEAN - the mean of the calibration block is used * MEDIAN - the median of the calibration block is used * INTERPOLATION - an interpolation is done and applied

type dCSRef : Difference of the calibration source got during flat field computation [INPUT, MANDATORY, default=no default value]

type threshold : threshold used to trigger an alert when dCsRef/dCS ratio is too big or too small [INPUT, MANDATORY, default=no default value]

type copy : boolean with the possible values : [INPUT, MANDATORY, default=no default value]

* false (jython: 0) - inFrames will be modified and returned * true (jython: 1) - a copy of inFrames will be returned

type quality : the quality context [INPUT, MANDATORY, default=no default value]

History

- 2007-12-20 BM: initial version
- 2008-02-08 BM: full version using Pal storage
- 2008-04-18 BM: comment pal storage section - only memory usage will be done - ekki will
- 2008-06-06 BM: SPR 888 - replaced the deprecated simplePool class by LocalPool interface
- 2008-07-16 RH: D_PACS_SPG_200_18
- 2008-07-24 BM: fix bug when signal size = 1
- 2008-10-04 BM: uses blocks previously built by PhotDiffCalTask
- 2008-10-09 BM: removed display messages
- 2008-10-16 BM: creates Noise dataset if it doesn't exist
- 2009-03-18 BM: works pixel by pixel and adds mask management. MEAN,MEDIAN,INTERPOLATE not checked and noise not computed
- 2009-24-27 BM: replace Master mask by UNUSABLE mask

1.197. photExposure

Full Name:	herschel.pacs.toolboxes.phot.photExposure
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.phot import photExposure

Description

Creates PACS photometer exposure maps for scan observations

API Summary

Jython Syntax
`ImageDataset=photExposure(angleArrayToMap, angleNorthToMap, scanLegLength, scanCrossScan, scanLegNum, scanSpeed, mapHomogeneous, mapSquare, mapRepetitions, mapType, filter)`

Property
`&quot;Red&quot;] filter = ["Blue1" ["Blue2", MANDATORY, default=no default value]`

API details

Property

`&quot;Red&quot;] filter = ["Blue1" ["Blue2", MANDATORY, default=no default value]`
angleArrayToMap = [double] angleNorthToMap = [double] scanLegLength = [double], (arcmin) scanCrossScan = [double], (arcsec) scanLegNum = [integer] scanSpeed = [double], (arcsec/sec) mapHomogeneous = [0/1] mapSquare = [0/1] mapRepetitions = [integer] mapType = [0=sensitivity, 1=exposure_time, 2=relative_time]

See also

- ???

History

- 1.0 10-Dec-2006 RV - Initial version
- 1.1 11-Mar-2007 RV - Extended parameter list
- Inter-matrix gaps
- 1.2 12-Mar-2007 RV - Optimized for faster performance

1.198. PhotFlagAdcSaturationTask

Full Name:	herschel.pacs.spg.phot.PhotFlagAdcSaturationTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.phot import PhotFlagAdcSaturationTask
Category:	pacs/spg/phot

Description

Jython task that sets the flags for saturated readouts

With QM model SATURATION mask is populated With FM and FS model, SATURATION_LOW and SATURATION_HIGH masks are poupulated This module is based on digital values (not converted values)

Example

Example 1: flag saturation

```
frames = photFlagAdcSaturation(frames)
```

API Summary

Jython Syntax

```
frames = photFlagAdcSaturation(frames, calTree=calTree,
satLimits=satLimits ,copy=copy)
```

Properties

frames [INPUT, Frames, default=MANDATORY]
String **calTree** [INPUT, OPTIONAL, default=default value null]
String **satLimits** [INPUT, OPTIONAL, default=default value null]
String **copy** [INPUT, OPTIONAL, default=default value : 0]

Limitations

No Limitation

API details

Properties

frames [INPUT, Frames, default=MANDATORY]

Input Frames, may be overwritten when copy=1

String calTree [INPUT, OPTIONAL, default=default value null]

Calibration Tree

String satLimits [INPUT, OPTIONAL, default=default value null]

Saturation limit Calibration data, calTree entries are overwritten)

String copy [INPUT, OPTIONAL, default=default value : 0]
Copy the input (0) and generate new output or overwrite the input (1)

See also

- ???

History

- 20051128 EkW initial version
- 20051208 EKW include comments of KO
- 20060317 EkW clean up documentation
- 20060807 BM add differential mode
- 20060830 JS task:mode ... replaced by type IN/OUT/IO
- 20060912 BM mask is created if it doesn't exist already
- 20060914 BM adapting code to new pcss syntax
- 20060914 BM minor changes
- 20061127 BM optimizing : memory space and speed
- 20070112 BM fixed bug
- 20070115 BM Added lower threshold detection
- 20070115 BM fixed bug
- 20070501 EkW In perform... check whether frame contains already SATURATION mask
- 20070501 EkW in perform_CQMFunc_photFlagSaturation fix rows etc. parameter
- 20071030 EkW move to new calibration framework / complete restructure
- 20080221 EkW adopt to new calibration framework
- 20080228 BM change outFrames.getStartTime() into outFrames.getStartDate()
- 20080429 EkW SPR 861
- 20080916 BM fixed bug for SPR 861 and SPR 1020 - let's see also cal product wrapper
- 20090217 BM renamed former task PhotFlagSaturation - generates ADC_SATURATION mask

1.199. PhotFlagBadPixelsTask

Full Name:	herschel.pacs.spg.phot.PhotFlagBadPixelsTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.phot import PhotFlagBadPixelsTask
Category:	pacs/spg/phot

Description

Jython task that sets the flags for bad pixels ("BAD_PIXEL" mask)

Bad pixels are defined by the PACS system engineers as pixels that either don't detect signals at all, pixels that have etended noise or any other unusual features that recommend only very careful use of their signal.

The bad pixels are defined in a calibration file with the name BadPixelMask. The contents of the calfile are converted into a mask with the same name (BadPixelMask) by this task. The mask appears in the input frames object after the task has been executed.

Bad pixels have to be distinguised from Blindpixels. In every Frames object there is also a BlindPixelMask. It contains the contents of the uplinked detector selection table. As a result PACS does not send any data for blind pixels. Bad pixels on the other hand, can only have extended noise and their data may still be downlinked.

So we can conclude: every Blindpixel is also a Badpixel, but not every Badpixel is a Blindpixel - although the overlap may be strong.

Example

Example 1: flag bad pixel

```
frames = photFlagBadPixels(frames)
```

API Summary

Jython Syntax

```
frames = photFlagBadPixels(frames, calTree = calTree,
badPixelMask=badPixelMask ,copy=copy)
```

Properties

Frames **frames** [IN/OUT, MANDATORY, default=NO default value]
PacsCal **calTree** [INPUT, OPTIONAL, default=default value :null]
BadPixelMask **badPixelMask** [INPUT, OPTIONAL, default=default value : null]
String **copy** [INPUT, OPTIONAL, default=default value : 0]

Limitations

No Limitation

API details

Properties

Frames frames [IN/OUT, MANDATORY, default=NO default value]
Input Frames, may be overwritten when copy=1

PacsCal calTree [INPUT, OPTIONAL, default=default value :null]
optional Calibration Tree

BadPixelMask badPixelMask [INPUT, OPTIONAL, default=default value : null]
optional the name of the BadPixelData calibration file

String copy [INPUT, OPTIONAL, default=default value : 0]
Copy the input (0) and generate new output or overwrite the input (1)

See also

- ???

History

- 20060728 BM initial version
- 20070903 BM SPR-0754 has been resolved
- 20071017 EKW Change default setting calibrafion file id
- 20071121 MW adapted cal/PhotBadPixelsMask and spg/PhotBadPixelsMask to new calibration system
- 20080219 EkW adopt again to the new calibration framework
- 20080228 BM change outFrames.getStartTime() into outFrames.getStartDate()
- 20091121 MW extended documentation

1.200. PhotFlagClSaturationTask

Full Name:	herschel.pacs.toolboxes.phot.PhotFlagClSaturationTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.phot import PhotFlagClSaturationTask
Category:	pacs/spg/phot

Description

Jython task that sets the flags for saturated readouts

This module base on digital values (not converted values) CL_SATURATION_LOW and CL_SATURATION_HIGH masks are populated by this task and attached to the frames.

Example

Example 1: flag saturation

```
frames = photFlagClSaturation(frames)
```

API Summary

Jython Syntax

```
frames = photFlagClSaturation(frames, hkdata=hk, calTree=calTree,
clSaturationLimits=clLimits ,clTransferFunction=clFunction,copy=copy)
```

Properties

type **frames** [INPUT, MANDATORY, default=no default value]
type **hk is a TableDataset issued from the ObservationContext and containing hk information** [INPUT, MANDATORY, default=no default value]
type **calTree is the calibration tree** [INPUT, MANDATORY, default=no default value]
type **clLimits contains high and low saturation limits of the CL** [INPUT, MANDATORY, default=no default value]
type **clFunction contains measurements given Vsignal = f(Vhblind** [Vbolo), MANDATORY, default=no default value]
type **copy forced to return a copy of frame when its value is 1** [INPUT, MANDATORY, default=no default value]
frames [INPUT, Frames, default=MANDATORY]
TableDataset **hk** [INPUT, MANDATORY, default=NO default value]
CalibrationTree **calTree** [INPUT, OPTIONAL, default=default value null]
ClSaturationLimits **clLimits** [INPUT, OPTIONAL, default=default value null]
CLTransferFunction **clTransferFunction** [INPUT, OPTIONAL, default=default value null]
String **copy** [INPUT, OPTIONAL, default=default value : 0]

Properties
type **frames containing bolstatus already sampling on science base time** [INPUT, MANDATORY, default=no default value]
type **verbose if true mode values are converted to a String** [INPUT, MANDATORY, default=no default value]

Limitations

No Limitation

API details

Properties

type frames [INPUT, MANDATORY, default=no default value]

type hk is a TableDataset issued from the ObservationContext and containing hk information [INPUT, MANDATORY, default=no default value]

type calTree is the calibration tree [INPUT, MANDATORY, default=no default value]

type clLimits contains high and low saturation limits of the CL [INPUT, MANDATORY, default=no default value]

type clFunction contains measurements given Vsignal = f(Vhblind [Vbolo), MANDATORY, default=no default value]

type copy forced to return a copy of frame when its value is 1 [INPUT, MANDATORY, default=no default value]

frames [INPUT, Frames, default=MANDATORY]

Input Frames, may be overwritten when copy=1

TableDataset hk [INPUT, MANDATORY, default=NO default value]

Housekeeping data on BOLC (issued form the ObservationContext)

CalibrationTree calTree [INPUT, OPTIONAL, default=default value null]

Calibration Tree

ClSaturationLimits clLimits [INPUT, OPTIONAL, default=default value null]

CL Saturation limit Calibration data, calTree entries are overwritten)

CLTransferFunction clTransferFunction [INPUT, OPTIONAL, default=default value null]

CL Transfer function Calibration data, calTree entries are overwritten)

String copy [INPUT, OPTIONAL, default=default value : 0]
Copy the input (0) and generate new output or overwrite the input (1)

type frames containing bolstatus already sampling on science base time [INPUT, MANDATORY, default=no default value]

type verbose if true mode values are converted to a String [INPUT, MANDATORY, default=no default value]

See also

- ???

History

- 20090217 BM first version
- 20090806 BM task has been completely rewrite. This version checks BOLC configuration and build intervals of stable configuration.
- Cl images saturation (top and bottom) are build by interval and checked by interval to make the finals top and bot masks

1.201. PhotFlagSaturationTask

Full Name:	herschel.pacs.spg.phot.PhotFlagSaturationTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.phot import PhotFlagSaturationTask
Category:	pacs/spg/phot

Description

Jython task that sets the flags for saturated readouts (populates SATURATION_LOW and SATURATION_HIGH mask).

This module is based on digital readout values (not converted values)

Example

Example 1: flag saturation

```
frames = photFlagSaturation(frames,calTree)            --- adc checking by
default
frames = photFlagSaturation(frames,calTree,hk,"cl")        --- cl checking only
frames = photFlagSaturation(frames,calTree,hk,"adc")        --- adc checking
only
frames = photFlagSaturation(frames,calTree,hk,"full")        --- cl & adc
checking
```

API Summary

Jython Syntax

```
frames = photFlagSaturation(frames,
calTree=calTree, hkdata=hk, check={adc,cl,full},
adcSaturation=satLimits,clSaturationLimits=clLimits ,clTransferFunction=clFunct
```

Properties

type **frames** [INPUT, MANDATORY, default=no default value]
type **calTree is the calibration tree** [INPUT, MANDATORY, default=no default value]
type **hk is a TableDataset issued from the ObservationContext and containing hk information** [INPUT, MANDATORY, default=no default value]
full } by default adc only **check = { adc** [cl, MANDATORY, default=no default value]
type **adcSaturation contains low and high bounds of the adc according to the BOLC mode selected** [INPUT, MANDATORY, default=no default value]
type **clLimits contains high and low saturation limits of the CL** [INPUT, MANDATORY, default=no default value]
type **clFunction contains measurements given Vsignal = f(Vhblind** [Vbolo), MANDATORY, default=no default value]
type **copy forced to return a copy of frame when its value is 1** [INPUT, MANDATORY, default=no default value]

Properties
frames [INPUT, Frames, default=MANDATORY]
TableDataset **hk** [INPUT, MANDATORY, default=NO default value]
CalibrationTree **calTree** [INPUT, OPTIONAL, default=default value null]
SatLimits **satLimits** [INPUT, OPTIONAL, default=default value is null]
ClSaturationLimits **clLimits** [INPUT, OPTIONAL, default=default value null]
ClTransferFunction **clTransferFunction** [INPUT, OPTIONAL, default=default value null]
String **copy** [INPUT, OPTIONAL, default=default value : 0]

Limitations

No Limitation

API details

Properties

type frames [INPUT, MANDATORY, default=no default value]

type calTree is the calibration tree [INPUT, MANDATORY, default=no default value]

type hk is a TableDataset issued from the ObservationContext and containing hk information [INPUT, MANDATORY, default=no default value]

full } by default adc only check = { adc [cl, MANDATORY, default=no default value]

type adcSaturation contains low and high bounds of the adc according to the BOLC mode selected [INPUT, MANDATORY, default=no default value]

type clLimits contains high and low saturation limits of the CL [INPUT, MANDATORY, default=no default value]

type clFunction contains measurements given Vsignal = f(Vhblind [Vbolo), MANDATORY, default=no default value]

type copy forced to return a copy of frame when its value is 1 [INPUT, MANDATORY, default=no default value]

frames [INPUT, Frames, default=MANDATORY]

Input Frames, may be overwritten when copy=1

TableDataset hk [INPUT, MANDATORY, default=NO default value]
Housekeeping data on BOLC (issued form the ObservationContext)

CalibrationTree calTree [INPUT, OPTIONAL, default=default value null]
Calibration Tree

SatLimits satLimits [INPUT, OPTIONAL, default=default value is null]
adcSaturation contains low and high bounds of the ADC according to the BOLC mode selected

ClSaturationLimits clLimits [INPUT, OPTIONAL, default=default value null]
CL Saturation limit Calibration data, calTree entries are overwritten)

ClTransferFunction clTransferFunction [INPUT, OPTIONAL, default=default value null]
CL Transfer function Calibration data, calTree entries are overwritten)

String copy [INPUT, OPTIONAL, default=default value : 0]
Copy the input (0) and generate new output or overwrite the input (1)

See also

- ???

History

- 20090217 BM first version

1.202. PhotGlobalDriftCorrectionTask

Full Name:	herschel.pacs.toolboxes.spg.PhotGlobalDriftCorrectionTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spg import PhotGlobalDriftCorrectionTask

Description

Photometer global drift correction

These routines are created to help with MADmap reductions. PLEASE READ These are not meant to be user friendly routines (yet). There are a number of assumption and quite possibly errors Please contact the author but don't expect smooth sailing!!! No testing, other than usage by author, has been done. You may/can/will easily break these routines. An alternative may be to simply wait for the first user release of these routines.

API Summary

Jython Syntax

```
Frames outFrames = photGlobalDriftCorrection(Frames inFrames[,
Integer model, Integer copy=copy, Integer binsize=binsize,
String datadir=datadir, String outprefix=outprefix, Boolean
doPlot=doPlot, Boolean slowMedian=slowMedian, Boolean
useMedian=useMedian, Boolean subFit=subFit, Integer order=order,
Integer verbose=verbose]
```

Properties

Frames **inFrames** [INPUT, MANDATORY, default=NO default value]
Integer **model** [INPUT, Optional, default=default value: 1]
Integer **copy** [INPUT, Optional, default=default value: 0]
Integer **binsize** [INPUT, Optional, default=default value: 1000]
String **datadir** [INPUT, Optional, default=default value: "./"]
String **outprefix** [INPUT, Optional, default=default value: "plot"]
Boolean **doPlot** [INPUT, Optional, default=default value: False]
Boolean **slowMedian** [INPUT, Optional, default=default value: False]
Boolean **useMedian** [INPUT, Optional, default=default value: False]
Boolean **subFit** [INPUT, Optional, default=default value: False]
Integer **order** [INPUT, Optional, default=default value: 3]
Integer **verbose** [INPUT, Optional, default=default value: 0]
Frames **outFrames** [OUT, Optional, default=NO default value]

API details

Properties

Frames inFrames [INPUT, MANDATORY, default=NO default value]

Input Frames object

Integer model [INPUT, Optional, default=default value: 1]

Determine which model to use for the global drift correction. model=1, the drift is modeled as a linear monotonic change in signal that is well-fit by a line. model=2, global median subtraction per pixel model=3, global median subtraction model=4, ???

Integer copy [INPUT, Optional, default=default value: 0]

Indicates if the new frames should be copied and returned (copy=1) or if the input frame is changed (copy=0)

Integer binsize [INPUT, Optional, default=default value: 1000]

The size of the bin to use for MIN/MEDIAN estimate

String datadir [INPUT, Optional, default=default value: "./"]

Specify where to store the plot.

String outprefix [INPUT, Optional, default=default value: "plot"]

Prefix to plot filename when saving.

Boolean doPlot [INPUT, Optional, default=default value: False]

Recommended. See the fit and save it if True.

Boolean slowMedian [INPUT, Optional, default=default value: False]

Compute the median by looping in jython. The alternative implementation is faster but crashes HIPE (SPR has been raised)

Boolean useMedian [INPUT, Optional, default=default value: False]

Use MEDIAN instead of MIN. Only applicable to data where most of the signal is sky not source.

Boolean subFit [INPUT, Optional, default=default value: False]

??? Subtract the fit if True

Integer order [INPUT, Optional, default=default value: 3]

???Order of the polynomial???

Integer verbose [INPUT, Optional, default=default value: 0]

Set verbose mode.

Frames outFrames [OUT, Optional, default=NO default value]

Returned Frames object

See also

- (not available publicly yet)

History

- 1.0 23-Nov-2009 BA Initial version of this Task

1.203. PhotHighpassFilterTask

Full Name:	herschel.pacs.spg.phot.PhotHighpassFilterTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.phot import PhotHighpassFilterTask

Description

High pass filtering of Photometer data

API Summary

Jython Syntax
`Frames outFrame = photHighpassFilter(Frames inFrame, int medianBox, int copy)`

Properties
`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
`int medianBox [INPUT, MANDATORY, default=default value: 0 (no median filtering)]`
`Integer copy [INPUT, Optional, default=default value: 0]`
`Frames outFrame [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

Frames inFrame [INPUT, MANDATORY, default=NO default value]
input Frames object

int medianBox [INPUT, MANDATORY, default=default value: 0 (no median filtering)]

Integer copy [INPUT, Optional, default=default value: 0]
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

Frames outFrame [OUTPUT, MANDATORY, default=NO default value]
returned Frames object containing the flags for frames not on the chopper plateaux

History

- 0.1 200709225 EkW initial version
- 1.0 20080306 EkW use CondenseBoxcar to speed up

1.204. photIltPqTool

Full Name:	herschel.pacs.toolboxes.phot.photIltPqTool
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.phot import photIltPqTool

Description

Given xy stage coordinates for single hole and chopper readback, predict

the pixel (=subarray) coordinates for the signal peak.

API Summary

Jython Syntax
`(p,q)=photIltPqTool(x,y,chopperPosition,camera[,zeroOffset=664] [,CalibrationTree calTree])`

Properties
`Double x [INPUT, MANDATORY, default=NO default value]`
`Double y [INPUT, MANDATORY, default=NO default value]`
`Int chopperPosition [INPUT, MANDATORY, default=NO default value]`
`String camera [INPUT, MANDATORY, default=NO default value]`
`Int zeroOffset [INPUT, OPTIONAL, default=default value: 664]`
`CalibrationTree calTree [INPUT, OPTIONAL, default=NO default value]`
`Double p [OUTPUT, MANDATORY, default=NO default value]`
`Double q [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Double x [INPUT, MANDATORY, default=NO default value]`

xy stage coordinate [mm]

`Double y [INPUT, MANDATORY, default=NO default value]`

xy stage coordinate [mm]

`Int chopperPosition [INPUT, MANDATORY, default=NO default value]`

Chopper readback position

`String camera [INPUT, MANDATORY, default=NO default value]`

"blue" or "red"

`Int zeroOffset [INPUT, OPTIONAL, default=default value: 664]`

chopper readback at optical zero, default value is for FM

CalibrationTree calTree [INPUT, OPTIONAL, default=NO default value]

The implicit default is FM, but with the ArrayInstrument calfile modified to version 2 which has values directly applicable for XY stage

Double p [OUTPUT, MANDATORY, default=NO default value]

subarray coordinate

Double q [OUTPUT, MANDATORY, default=NO default value]

subarray coordinate

History

- 20060710 DL first version
- 20080306 DL new cal framework
- 20081014 DL path to calfile adapted to PCSS change
- 20081027 WDM dp-pacs
- 20081125 DL paths yet again, more dp-pacs adjustments

1.205. photIltXyTool

Full Name:	herschel.pacs.toolboxes.phot.photIltXyTool
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.phot import photIltXyTool

Description

Given subarray pixel coordinates p,q and chopper readback, predict where

the xy stage with single hole has to be to have signal peak on that pixel

API Summary

Jython Syntax
`(x,y)=photIltXyTool(p,q,chopperPosition,camera[,zeroOffset=664] [,CalibrationTree calTree])`

Properties
`Double p [INPUT, MANDATORY, default=NO default value]`
`Double q [INPUT, MANDATORY, default=NO default value]`
`Int chopperPosition [INPUT, MANDATORY, default=No default value]`
`String camera [INPUT, MANDATORY, default=NO default value]`
`Int zeroOffset [INPUT, OPTIONAL, default=default value: 664]`
`CalibrationTree calTree [INPUT, OPTIONAL, default=NO default value]`
`Double x [OUTPUT, MANDATORY, default=NO default value]`
`Double y [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Double p [INPUT, MANDATORY, default=NO default value]`

pixel coordinate

`Double q [INPUT, MANDATORY, default=NO default value]`

pixel coordinate

`Int chopperPosition [INPUT, MANDATORY, default=No default value]`

Chopper readback position

`String camera [INPUT, MANDATORY, default=NO default value]`

"blue" or "red"

`Int zeroOffset [INPUT, OPTIONAL, default=default value: 664]`

chopper readback at optical zero, default value is for FM

CalibrationTree calTree [INPUT, OPTIONAL, default=NO default value]

The implicit default is FM, but with the ArrayInstrument calfile modified to version 2 which has values drectly applicable for XY stage

Double x [OUTPUT, MANDATORY, default=NO default value]

xy stage coordinate

Double y [OUTPUT, MANDATORY, default=NO default value]

xy stage coordinate

History

- 20060706 DL first version
- 20080306 DL new cal framework
- 20081014 DL path to calfile adapted to PCSS change
- 20081027 WDM pcss to dp-pacs
- 20081125 DL paths yet again, more dp-pacs adjustments

1.206. PhotMakeDithPosTask

Full Name:	herschel.pacs.spg.phot.PhotMakeDithPosTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.phot import PhotMakeDithPosTask

Description

Identify dither positions and put it into status word DithPos

API Summary

Jython Syntax
`Frames outFrame = photMakeDithPos(Frames inFrame, int copy)`

Properties
`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
`Integer copy [INPUT, Optional, default=default value: 0]`
`Frames outFrame [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
input Frames object

`Integer copy [INPUT, Optional, default=default value: 0]`
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

`Frames outFrame [OUTPUT, MANDATORY, default=NO default value]`
returned Frames object containing the flags for frames not on the chopper plateaux

History

- 0.5 20070801 EkW initial version
- 1.0 20080905 EkW SPR for non dithered observations
- 2.0 20081126 EkW SPR 1087 - Dith pos even for non dithered ...

1.207. PhotMakeRasPosCountTask

Full Name:	herschel.pacs.spg.phot.PhotMakeRasPosCountTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.phot import PhotMakeRasPosCountTask

Description

Identify raster positions and write a counter to the Status word

API Summary

Jython Syntax
`Frames outFrame = photMakeRasPosCount(Frames inFrame, int copy)`

Properties
`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
`Integer copy [INPUT, Optional, default=default value: 0]`
`Frames outFrame [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
input Frames object

`Integer copy [INPUT, Optional, default=default value: 0]`
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

`Frames outFrame [OUTPUT, MANDATORY, default=NO default value]`
returned Frames object containing the raster position counter in the Status

History

- 0.5 20070816 EkW initial version
- 1.0 20080830 EkW usage of Poiting Product

1.208. PhotModuleDriftCorrectionTask

Full Name:	herschel.pacs.toolboxes.spg.PhotModuleDriftCorrectionTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spg import PhotModuleDriftCorrectionTask

Description

Photometer module to module drift correction

These routines are created to help with MADmap reductions. This function applies the module to module drift correction on the PACS data. It does this by comparing the MEDIAN values of the modules to a reference module, here hardcoded as module #5. WARNINGS: (1): Only use it on the blue channel data. (2): Use it after the pipeline step which converts the output to volts. PLEASE READ These are not meant to be user friendly routines (yet). There are a number of assumption and quite possibly errors Please contact the author but don't expect smooth sailing!!! No testing, other than usage by author, has been done. You may/can/will easily break these routines. An alternative may be to simply wait for the first user release of these routines.

API Summary

Jython Syntax
`Frames outFrames = photModuleDriftCorrection(Frames inFrames[, copy=copy, datadir=datadir, outprefix=outprefix, doPlot=doPlot, q1=q1, q2=q2])`

Properties
`Frames` **`inFrames`** `[INPUT, MANDATORY, default=NO default value]`
`Integer` **`copy`** `[INPUT, Optional, default=default value: 0]`
`String` **`datadir`** `[INPUT, Optional, default=default value: "./"]`
`String` **`outprefix`** `[INPUT, Optional, default=default value: "moduleDriftCorrection"]`
`Boolean` **`doPlot`** `[INPUT, Optional, default=default value: False]`
`Double` **`q1`** `[INPUT, Optional, default=default value: -0.1]`
`Double` **`q2`** `[INPUT, Optional, default=default value: 0.1]`
`Frames` **`outFrames`** `[OUT, Optional, default=NO default value]`

API details

Properties

`Frames inFrames [INPUT, MANDATORY, default=NO default value]`
Input Frames object

`Integer copy [INPUT, Optional, default=default value: 0]`
Indicates if the new frames should be copied and returned (copy=1) or if the input frame is changed (copy=0)

String datadir [INPUT, Optional, default=default value: "./"]

Directory where plots are saved

String outprefix [INPUT, Optional, default=default value: "moduleDriftCorrection"]

Output plot file prefix

Boolean doPlot [INPUT, Optional, default=default value: False]

Plot the fit and save the plot (as jpg) if True, otherwise no plotting

Double q1 [INPUT, Optional, default=default value: -0.1]

Lower limit of sky values (including the drift)

Double q2 [INPUT, Optional, default=default value: 0.1]

Upper limit of sky values (including the drift)

Frames outFrames [OUT, Optional, default=NO default value]

Returned Frames object

See also

- (not available publicly yet)

History

- 1.0 23-Nov-2009 BA Initial version of this Task

1.209. PhotOffsetCorrTask

Full Name:	herschel.pacs.toolboxes.phot.PhotOffsetCorrTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.phot import PhotOffsetCorrTask

Description

Offset correction of photometer data - part of preprocess for MadMap -

by either median removal or zero level calibration value subtraction

API Summary

Jython Syntax
`Frames outFrame = photOffsetCorr(Frames inFrame, int copy, int zeroLevelCorr, Selection regionSelection)`

Properties
`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
`Integer copy [INPUT, Optional, default=default value: 0]`
`Integer zeroLevelCorr [INPUT, Optional, default=default value: 0]`
`Selction regionSelection [INPUT, Optional, default=default value: None]`
`Frames outFrame [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Frames inFrame [INPUT, MANDATORY, default=NO default value]`

input Frames object

`Integer copy [INPUT, Optional, default=default value: 0]`

indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

`Integer zeroLevelCorr [INPUT, Optional, default=default value: 0]`

do median removal (zeroLevelCorr=0) or do zero level calibration value subtraction (zeroLevelCorr=1)

`Selction regionSelection [INPUT, Optional, default=default value: None]`

do median removal using the given selection If both zeroLevelCorr and regionSelection are specified and zeroLevelCorr==1, then zero level calibration value substraction is done.

`Frames outFrame [OUTPUT, MANDATORY, default=NO default value]`

returned Frames object containing Signal which has Median subtracted

History

- 1.1 10-21-2008 CL initial version

1.210. photPq2Uv

Full Name:	herschel.pacs.toolboxes.phot.photPq2Uv
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.phot import photPq2Uv

Description

Convert a single pair of bolometer subarray coordinates p,q to array

coordinates u,v Convert the integer subarray coordinates of a pixel, or interpolated coordinates, to array coordinates.

API Summary

Jython Syntax
`(u,v) = photPq2Uv(p,q,camera [,model="FM"] [,scope="BASE"] [,CalibrationTree calTree])`

Properties
`Double p [INPUT, MANDATORY, default=NO default value]`
`Double q [INPUT, MANDATORY, default=NO default value]`
`String camera [INPUT, MANDATORY, default=NO default value]`
`String model [INPUT, OPTIONAL, default=default value: "FM"]`
`String scope [INPUT, OPTIONAL, default=default value: "BASE"]`
`CalibrationTree calTree [INPUT, OPTIONAL, default=NO default value]`
`Double u [OUTPUT, OPTIONAL, default=NO default value]`
`Double v [OUTPUT, OPTIONAL, default=NO default value]`

Miscellaneous

This jython function is intended for cal analysis procedures, manual sanity checks etc., but not for bulk calibration of PACS data which can be done more efficiently by directly using the calfile. For non-integer coordinates a simple bilinear interpolation is done which is not very meaningful in the inter-matrix gaps...

API details

Properties

`Double p [INPUT, MANDATORY, default=NO default value]`

subarray coordinate of pixel

`Double q [INPUT, MANDATORY, default=NO default value]`

subarray coordinate of pixel

`String camera [INPUT, MANDATORY, default=NO default value]`

"blue" or "red"

String model [INPUT, OPTIONAL, default=default value: "FM"]

String scope [INPUT, OPTIONAL, default=default value: "BASE"]

CalibrationTree calTree [INPUT, OPTIONAL, default=NO default value]

if a caltree is supplied, this will override the model and scope settings

Double u [OUTPUT, OPTIONAL, default=NO default value]

array coordinate

Double v [OUTPUT, OPTIONAL, default=NO default value]

array coordinate

See also

- ???
- ???

History

- 24/06/2005 DL Creation
- 04/08/2005 JS header adopted to jtags syntax
- 05/07/2006 DL fixed eval syntax
- 05/03/2008 DL allow non-integer input coordinates, new cal framework
- 03/12/2008 DL replace deprecated Interpolation2D by Bilinear

1.211. PhotProjectPointSourceTask

Full Name:	herschel.pacs.spg.phot.PhotProjectPointSourceTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.phot import PhotProjectPointSourceTask
Category:	pacs/spg/phot

Description

Project Photometer PointSource image

Define a WCS for an image that is

* centered on the hspot requested coordinate (obs context meta data)

* tilted by the position angle (status)

* total width a little more than 5 blue matrices

* total height a little more than 3 matrices

* pixel size tbd, no bigger than a blue pixel, I'd guess I'd use 1" but this is just a guess.

This will outline an image covering the total footprint of a point source AOR

1) For each of the 3 (or 1) double differential images

- run assignradec and photproject 4 times for this one frame and all 4

combinations of (ra+dec) and chopper that are applicable to this dither

- For the two combination that correspond to negative beams, multiply the

flux of the result by -1

After this step you should have 12 (or 4) simple images, all on exactly

the same WCS, all with a positive beam at the HSPOT position, but covering

different parts of the total image and with different patterns of

secondary beams.

2) Do a weighted average of these 12 or 4 images. For general distrust in

'our' system and since they are on the same pixel grid already, I would

perhaps not use fancy sara functionality but do an own quick weighted

average pixel by pixel. In the long term, this should use the error data

from the frames and the photproject result.

In the short term, using the coverage map should be perfectly

fine if we reassure ourselfes that it is properly made by photproject in

this case

API Summary

Jython Syntax
`frames = projectPhotometerPointSource(frames, ppointing [raCent, decCent, width, height,rotAngle, outputixelsize, calibration, allInOne])`

Properties
Frames **frames** [IN, MANDATORY, default=Frames in]
TabbleDataset **ppointings** [IN, MANDATORY, default=pointinmg informationm per chop / nod position]
SimpleImage **image** [OUT, MANDATORY, default=no default value]

API details

Properties

Frames frames [IN, MANDATORY, default=Frames in]

TabbleDataset ppointings [IN, MANDATORY, default=pointinmg informationm per chop / nod position]

SimpleImage image [OUT, MANDATORY, default=no default value]

1.212. PhotProjectTask

Full Name:	herschel.pacs.spg.phot.PhotProjectTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import PhotProjectTask

Description

This task accepts a frames object as input that has been processed by a pacs pipeline and maps the signal data into a sky map.

The minimum contents of the frames object must be a signal array and the pointing status entries provided by the AddInstantPointingTask. PhotProject maps the images contained in the frames class into one single map and returns it as a SimpleImage object that contains the map, the coordinates (in form of a world coordinate system wcs) and inputexposure, noise (error) and weight maps.

The mapping process by default uses all activated masks from the input frames object.

A second functionality of PhotProject is the preparation for the second level deglitching. This is done with the paramter deglitch = true. In that case, a MapIndex Object is created (instead of the SimpleImage) that contains the necessary information to deglitch the signal contributions into each output pixel with a Sigma clipping. This is done with the IIndLevelDeglitchTask, which needs the MapIndex as input.

Example

Example 1: from Jide:

```
from herschel.pacs.spg import PhotProjectTask
photProject = PhotProjectTask()
map = photProject(frames)
photProject(frames, deglitch=True, slimindex = True)
mapindex = photProject.getValue("index")
<br>
```

API Summary

Properties
MANDATORY **inframes: INPUT** [Frames, no default value., default=no default value]
OPTIONAL **calTree: INPUT** [PacsCal, default is the system caltree., default=no default value]
OPTIONAL **copy: INPUT** [Integer, default is 0 (= false)., default=no default value]
OPTIONAL **weightedsignal INPUT** [Boolean, default is False., default=no default value]
OPTIONAL **outputPixelsize: INPUT** [DOUBLE, default is the same value as the inframes pixelsize (3.2 arcsecs for the blue photometer, default=6.4 for red).]
OPTIONAL **monitor: INPUT** [Boolean, default is False., default=no default value]
OPTIONAL **calibration: INPUT** [Boolean, default is True., default=no default value]

Properties
OPTIONAL **optimizeOrientation: INPUT** [Boolean, default is False., default=no default value]
OPTIONAL **minRotation: INPUT** [Integer, default value is 15., default=no default value]
OPTIONAL **wcs: INPUT** [Wcs, no default value., default=no default value]
OPTIONAL **mapcoordinates: INPUT** [Double1d, no default value., default=no default value]
OPTIONAL **deglitch: INPUT** [Boolean, default value is False., default=no default value]
OPTIONAL **slimindex: INPUT** [Boolean, default value is True., default=no default value]
OPTIONAL **image: OUTPUT** [SimpleImage, no default value., default=no default value]
OPTIONAL **index: OUTPUT** [MapIndex, no default value., default=no default value]

API details

Properties

MANDATORY inframes: INPUT [Frames, no default value., default=no default value]

the input frames object

OPTIONAL calTree: INPUT [PacsCal, default is the system caltree., default=no default value]

CalibrationTree for calfile access

OPTIONAL copy: INPUT [Integer, default is 0 (= false)., default=no default value]

0 if the original frames class should be used, 1 if a copied version should be use.

OPTIONAL weightedsignal INPUT [Boolean, default is False., default=no default value]

set this to true, if the signal contributions to the map pixels should be weighted with the signal error. The error is taken from the noise array in the inframes object.

OPTIONAL outputPixelsize: INPUT [DOUBLE, default is the same value as the inframes pixelsize (3.2 arcsecs for the blue photometer, default=6.4 for red).]

the size of a pixel in the output dataset in arcseconds. Default is the same size as the input (6.4 arcsecs for the red and 3.2 arcsecs for the blue photometer).

OPTIONAL monitor: INPUT [Boolean, default is False., default=no default value]

This is an engineering option only. If true, shows the map monitor that allows in situ monitoring of the map building process. Default value: false.

Please note, that the MapMonitor dramatically increases the memory requirements.

OPTIONAL calibration: INPUT [Boolean, default is True., default=no default value]

decides, whether the spatial calibration files should be used to calculate pixel geometries

(calibration = True), or whether the detector pixels are assumed to be quadratic with the textbook sizes (calibration = False). Default value: true.

If false, the frames object must contain Ra/Dec coordinate arrays.

OPTIONAL optimizeOrientation: INPUT [Boolean, default is False., default=no default value]

rotates the map by an angle between minRotation and 89 degrees in a way that the coverage of the outputimage is maximized

as a result, north points no longer upwards.

Possible values False (default): no automatic rotation, True: automatic rotation

OPTIONAL minRotation: INPUT [Integer, default value is 15., default=no default value]

defines the minimum angle for which optimizeOrientation should be applied. Default value is 15 degrees. If optimizeOrientation finds out that the angle is below this value, no rotation will be done. A message is logged instead.

OPTIONAL wcs: INPUT [Wcs, no default value., default=no default value]

allows to specify a customized wcs that is used for projection.

Usually photProject calculates and optimizes its own Wcs. The wcs parameter allows to overwrite the internally created wcs. The easiest way to create a Wcs is to use the Wcs4mapTask (that is also used internally by this task) and modify its parameters. The necessary paramters to modify are:

wcs.setCrota2(angle) : the rotation angle in decimal degrees (like 45.0 for 45 degrees)

wcs.setCrpix1(crpix1) :The reference pixel position of axis 1. Use the center of your map: mapwidth/2 (the Ra-coordiante)

wcs.setCrpix2(crpix2) :The reference pixel position of axis 2. Use the center of your map: mapheight/2 (the Dec-coordinate)

wcs.setCrval1(crval1) :The coordinate of crpix1 (ra-value in decimal degrees - use Maptools.ra_2_decimal(int hours, int arcmin, double arcsec) for conversion)

wcs.setCrval2(crval2) :The coordinate of crpix2 (dec-value in decimal degrees - use Maptools.dec_2_decimal(int degree, int arcmin, double arcsec) for conversion)

wcs.setParameter("NAXIS1", mapwidth, "Number of Pixels along axis 1.") :mapwidth = crpix1*2, if you follow these instructions

wcs.setParameter("NAXIS2", mapheight, "Number of Pixels along axis 2") :mapheight = crpix2*2, if you follow these instructions

OPTIONAL mapcoordinates: INPUT [Double1d, no default value., default=no default value]

This paramter is deprecated and will be removed. Please use the parameter wcs instead.

OPTIONAL deglitch: INPUT [Boolean, default value is False., default=no default value]

specifies, that PhotProject does not create a map, but writes all contributions to the map into a MapIndex object instead.

After PhotProject is finsished, the MapIndex can be obtained with mstack = photProject.getValue("index").

The PacsMapstack is the input object for the second level deglitching with IIndLevelDeglitchingTask.

Possible values: False (default) or True.

OPTIONAL slimindex: INPUT [Boolean, default value is True., default=no default value]

together with deglitch = True instructs PhotProject to build a memory efficient index for deglitching.

Building an slimindex means that second level deglitching will take longer, but can be processed with significantly less memory requirements (ca. 20% compared to slimindex = false).

OPTIONAL image: OUTPUT [SimpleImage, no default value., default=no default value]

the returned SimpleImage that contains a map and a Wcs (World Coordinate System).

OPTIONAL index: OUTPUT [MapIndex, no default value., default=no default value]

if the option deglitch is used, the SimpleImage is not created. Instead a MapIndex object is produced that must be used as input to the IIndLevelDeglitchingTask. The index is not returned by index = photProject(....) like the SimpleImage. Instead, photProject has to be executed (photProject(...) and then index = photProject.getValue("index") must be called.

1.213. PhotReadMaskFromImageTask

Full Name:	herschel.pacs.spg.phot.PhotReadMaskFromImageTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import PhotReadMaskFromImageTask

Description

This task takes a mask that is defined within a SimpleImage and puts the masked coordinates back

into an equivalent mask of the Frames object. The Frames coordinates must have strong overlap with the SimpleImages Wcs.

A typical usecase is this:

In a first run, a map has been created from a Frames object. In this map, the sources are found and masked. Then this task will write the masked source locations back into the Frames object.

Attention:

A note for usage. The mask that is created by this task may intentionally mask your sources. Please keep that in mind and do not forget to remove the mask for further processing steps, where it may influence your results in unintended ways.

API Summary

Jython Syntax
`Frames outFrame = PhotReadMaskFromImage(Frames inframes, SimpleIMage si [, path = "./mymask.fits"]`
`[, maskname = "MyMask"] [, copy = True]`
`[, caltree = None][,subArrayArray = MyVersionOfSubArrayArray]`
`[, arrayInstrument = MyVersionOfArrayInstrument])`

Properties
`Frames` **`inframes`** `[INPUT, MANDATORY, default=NO default value]`
`SimpleIMage` **`si`** `[INPUT, OPTIONAL, default=NO default value]`
`String` **`path`** `[INPUT, OPTIONAL, default=NO default value]`
`String` **`maskname`** `[INPUT, OPTIONAL, default=default value: "Highpassmask"]`
`Boolean` **`extendedMasking`** `[INPUT, OPTIONAL, default=default value: False]`
`Double` **`threshold`** `[INPUT, OPTIONAL, default=default value: 0.5]`
`Boolean` **`copy`** `[INPUT, OPTIONAL, default=default value: False]`
`PacsCal` **`caltree`** `[INPUT, OPTIONAL, default=default value: the latest version that is provided by the system]`
`OPTIONAL` **`subArrayArray INPUT`** `[SubArrayArray, default value: the latest version that is provided by the system, default=no default value]`
`OPTIONAL` **`arrayInstrument INPUT`** `[ArrayInstrument, default value: the latest version that is provided by the system, default=no default value]`

API details

Properties

Frames inframes [INPUT, MANDATORY, default=NO default value]

input Frames object

SimpleIMage si [INPUT, OPTIONAL, default=NO default value]

a SimpleImage that contains a Wcs and an image array. In the image array, everything >=threshold is interpreted as masked, all other values as unmasked!

String path [INPUT, OPTIONAL, default=NO default value]

Alternatively to the SimpleImage object si, a path to a fits file that contains a Simple image with the specified properties of si can be provided. If a SimpleImage AND a path is provided, the SimpleImage will be processed and the path ignored.

String maskname [INPUT, OPTIONAL, default=default value: "Highpassmask"]

the name of the new mask that is created in the inframes object.

Boolean extendedMasking [INPUT, OPTIONAL, default=default value: False]

in some cases, for example if you have many one pixel only maskpoints, the mask cannot be perfectly written back into the frames object. This is because a frames pixel is projected into several map pixels. If only one map pixel is masked, writing it back into the frames object may be inaccurate. In these cases the extension of the masked region may solve the problem or this option, that extends the frames mask to neighbouring pixels. Default value: false

Double threshold [INPUT, OPTIONAL, default=default value: 0.5]

the limit for identifying a a value as masked. Values >=threshold are interpreted as masked, all other values as unmasked!

Boolean copy [INPUT, OPTIONAL, default=default value: False]

copies the inframes object before processing (copy = true) or not (copy = false)

PacsCal caltree [INPUT, OPTIONAL, default=default value: the latest version that is provided by the system]

A PacsCal object. It is used for calculation of Frames pixel Ra and Dec values.

OPTIONAL subArrayArray INPUT [SubArrayArray, default value: the latest version that is provided by the system, default=no default value]

A subArrayArray object. It is used for calculation of Frames pixel Ra and Dec values.

OPTIONAL arrayInstrument INPUT [ArrayInstrument, default value: the latest version that is provided by the system, default=no default value]

An arrayInstrument object. It is used for calculation of Frames pixel Ra and Dec values.

History

- 1.0 20090612 MW initial version

- 1.1 20090721 MW extended masking option

1.214. PhotRemoveMedianTask

Full Name:	herschel.pacs.toolboxes.phot.PhotRemoveMedianTask
Type:	Java Task -
Import:	from herschel.pacs.toolboxes.phot import PhotRemoveMedianTask
Category:	class

Description

This task is used with scan MAp AOR, where no chopping exists

The goal is to remove one part of the offset before applying scanMap This goal is reached by the extraction of the median of the cube, then by subtraction of this median

API Summary

Jython Syntax
`outFrames = PhotRemoveMedian(inFrames [,calTree=calTree] [,quality=quality][,copy=copy])`

Properties
`type` **`inFrames : the input frame object coming from the level 0`** `[INPUT, MANDATORY, default=no default value]`
`type` **`calTree : valide calibration tree`** `[INPUT, MANDATORY, default=no default value]`
`type` **`copy : boolean with the possible values :`** `[INPUT, MANDATORY, default=no default value]`
`type` **`quality : the quality context`** `[INPUT, MANDATORY, default=no default value]`

API details

Properties

`type inFrames : the input frame object coming from the level 0 [INPUT, MANDATORY, default=no default value]`

`type calTree : valide calibration tree [INPUT, MANDATORY, default=no default value]`

`type copy : boolean with the possible values : [INPUT, MANDATORY, default=no default value]`

* false (jython: 0) - inFrames will be modified and returned * true (jython: 1) - a copy of inFrames will be returned

`type quality : the quality context [INPUT, MANDATORY, default=no default value]`

History

- 2008-04-10 - BM: : initial version

1.215. PhotRespFlatfieldCorrectionTask

Full Name:	herschel.pacs.spg.phot.PhotRespFlatfieldCorrectionTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import PhotRespFlatfieldCorrectionTask

Description

Converts Volt signal to Jansky and applies flat correction

Responsivity and flatfield calibration products are used during this step

Detail of the processing :

```
signal = signal / responsivity*flat
```

**Remark:

Except with the responsivity calibration product version 3, drift correction should be run when the bias settings in the observation and calibration data are different.

When a difference among these bias settings is found an error message is logged

Example

Example 1: from Jide

```
frames = photRespFlatfieldCorrection(frames) --- the most recent caltree is
used
frames = photRespFlatfieldCorrection(frames,calTree=calTree) --- specific
calTree is used
frames = photRespFlatfieldCorrection(frames,calTree=calTree,copy=1) --- copy
is done
```

API Summary

Jython Syntax

```
outFrame = PhotRespFlatfieldCorrection(inFrame [,calTre=calTree]
[,flatField=flatField] [responsivity=responsivity[,copy=copy])
```

Properties

Frames **inFrame** [INPUT, MANDATORY, default=no default value.]

PacsCal **calTree** [INPUT, Optional, default=default value: null]

FlatField **flatfield** [INPUT, Optional, default=no default value]

Responsivity **responsivity** [INPUT, Optional, default=no default value]

Integer **copy** [INPUT, Optional, default=default value: 0]

MANDATORY **outFrame** [Frames, No default value, default=no default value]

API details

Properties

Frames inFrame [INPUT, MANDATORY, default=no default value.]

is an input frame object ; for point source AOT, this object contains stack of images at dither position. All ON-OFF chopping data has been identified, substracted each other and averaged.

PacsCal calTree [INPUT, Optional, default=default value: null]

access to PACS calibration tree

FlatField flatfield [INPUT, Optional, default=no default value]

contains the flat field correction

Responsivity responsivity [INPUT, Optional, default=no default value]

contains conversion coefficients Volt to Jansky

Integer copy [INPUT, Optional, default=default value: 0]

indicates if the input frames is modified and returned (copy=0) or copied before changes (copy=1)

MANDATORY outFrame [Frames, No default value, default=no default value]

frames with Jansky signal values.

History

- 20072012 BM 0.5 initial version
- 20080106 EkW 1.0 adopted to pipeline framework, task framework, calibration access, coding standard, clean up
- 20080107 EkW 1.1 correct calfile access
- 20080128 BM 1.2 add noise
- 20080204 BM 1.3 link with the new calibration products (Flatfield and Responsivity)
- 20082002 BM 1.4 adds getFormat version and removes readCal function
- 20082002 EkW 1.5 clean up,task framework and implement calfile access
- 20080402 BM 1.6 bad pixels of the frames are set to NaN
- 20081016 BM creates Noise dataset if it doesn't exist
- 20090318 BM xxx this take takes in charge mask and works pixel by pixel
- 20092427 BM 1.20 replace Master mask by UNUSABLE mask
- 20101201 BM 1.27 bias control and urm doc

1.216. PhotSetSignalTask

Full Name:	herschel.pacs.toolboxes.phot.PhotSetSignalTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.phot import PhotSetSignalTask

Description

For test only. Filling signal with a special pattern

API Summary

Jython Syntax
`photPhotSetSignal(Frames InOutFrames,value[,altvalue] [,calVersion=calVersion])`

Properties
`InOutFrames` `[INPUT, Frames, default=MANDATORY]`
`String` **`calVersion`** `[INPUT, OPTIONAL, default=default value :CQM_1_0]`
`String` **`copy`** `[INPUT, OPTIONAL, default=default value : 0]`

API details

Properties

`InOutFrames [INPUT, Frames, default=MANDATORY]`
value is used to fill the background with this specific value altvalue is used to fill the foreground with this specific value

`String calVersion [INPUT, OPTIONAL, default=default value :CQM_1_0]`
Calibration Version e.g. CQM_2_0

`String copy [INPUT, OPTIONAL, default=default value : 0]`
Copy the input (0) and generate new output or overwrite the input (1)

History

- 1.1 20060807 BM initial version
- 1.2 20060830 JS Tasks: mode = ... replaced by type = IN/OUT/IO
- 1.3 20060912 BM minor changes
- 1.4 20070115 Added foreground and background concept

1.217. PhotShiftDithTask

Full Name:	herschel.pacs.spg.phot.PhotShiftDithTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.phot import PhotShiftDithTask

Description

Resample the Signals and shift the Dither positions into each other

API Summary

Jython Syntax
`SimpleImage image = photShiftDith(Frames inFrame)`

Properties
`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
`SimpleImage image [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
input Frames object

`SimpleImage image [OUTPUT, MANDATORY, default=NO default value]`

History

- 0.1 200708010 EkW initial version
- 0.2 20070813 EkW use proper value for Red Array
- 1.0 20090211 EkW Return SimpleImage rather frames SPR 1291

1.218. PhotTreatCSTask

Full Name:	herschel.pacs.toolboxes.phot.PhotTreatCSTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.phot import PhotTreatCSTask
Category:	pacs/toolboxes/spg

Description

Treat Calibration Blocks for flux calibration and trend analysis

Remove the calibration blocks afterwards.

API Summary

Jython Syntax
`framesOut = photTreatCS(framesIn, photHk, [,calTree] [,obs] [,keepCS])`

Properties
`type` **`inFrames Frames input class`** `[INPUT, MANDATORY, default=no default value]`
`type` **`photHk photometer housekeeping data`** `[INPUT, MANDATORY, default=no default value]`
`type` **`calTree Calibration tree`** `[optional, MANDATORY, default=no default value]`
`type` **`obs ObservationContext to attach TA Product`** `[optional, MANDATORY, default=no default value]`
`type` **`keepCS Keep the Calibration Blocks`** `[INPUT, MANDATORY, default=no default value]`

API details

Properties

type inFrames Frames input class [INPUT, MANDATORY, default=no default value]

type photHk photometer housekeeping data [INPUT, MANDATORY, default=no default value]

type calTree Calibration tree [optional, MANDATORY, default=no default value]

type obs ObservationContext to attach TA Product [optional, MANDATORY, default=no default value]

type keepCS Keep the Calibration Blocks [INPUT, MANDATORY, default=no default value]

History

- 20000923 EkW initial version basing on BMs pacsphotpointsource0

1.219. PhotTrendCSProduct

Full Name:	herschel.pacs.signal.PhotTrendCSProduct
Type:	Java Class -
Import:	from herschel.pacs.signal import PhotTrendCSProduct

History

- 2009-01-22 : (bm) initial version

1.220. PhotTrendCSProducts

Full Name:	herschel.pacs.signal.PhotTrendCSProducts
Type:	Java Class -
Import:	from herschel.pacs.signal import PhotTrendCSProducts

History

- 2009-01-22 : (bm) initial version

1.221. PhotTrendCSTask

Full Name:	herschel.pacs.spg.phot.PhotTrendCSTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import PhotTrendCSTask
Category:	class

API Summary

Jython Syntax
`outFrames = photTrendCS(inFrames, trend, hkdata [,copy=0|1])`
`outFrames = photTrendCS(inFrames, trend, seq=seq [,copy=0|1])`

Properties
type **inFrames : the input frame object containing the average of the images per plateau.** [INPUT, MANDATORY, default=no default value]
type **trend : a container of PhotTrendCSProduct objects to fill by this task** [INPUT, MANDATORY, default=no default value]
type **hkdata : is a TableDataset containing housekeeping data related to inFrames** [INPUT, MANDATORY, default=no default value]
type **seq : is a PacketSequence containing housekeeping data related to inFrames** [INPUT, MANDATORY, default=no default value]
type **copy : boolean with the possible values :** [INPUT, MANDATORY, default=no default value]

API details

Properties

type inFrames : the input frame object containing the average of the images per plateau. [INPUT, MANDATORY, default=no default value]

type trend : a container of PhotTrendCSProduct objects to fill by this task [INPUT, MANDATORY, default=no default value]

type hkdata : is a TableDataset containing housekeeping data related to inFrames [INPUT, MANDATORY, default=no default value]

type seq : is a PacketSequence containing housekeeping data related to inFrames [INPUT, MANDATORY, default=no default value]

type copy : boolean with the possible values : [INPUT, MANDATORY, default=no default value]

false (jython: 0) - inFrames is used and can be modified by this task true (jython: 1) - a copy of inFrames is done, inFrames can't be modified by this task

History

- `2008-03-12 - BM`: : initial version
- $Log: PhotTrendCSTask.java,v $
- Revision 1.1 2010/02/11 13:38:44 wim
- Code reorganisation
- Revision 1.1 2010/02/11 10:36:13 wim
- Code reorganisation
- Revision 1.14 2010/02/02 11:34:21 hsclib
- LGPL
- Revision 1.13 2010/01/28 16:38:42 juergen
- SPR 999
- in case of error: throws RuntimeException
- Revision 1.12 2009/11/06 08:30:06 bmorin
- PACS-2133
- Revision 1.11 2009/08/06 13:05:40 bmorin
- SPR: PACS-1871 and PACS-1917: ICS resistance converted in temperature
- Revision 1.10 2009/05/06 16:04:05 bmorin
- SPR 1539 - cleaning message
- Revision 1.9 2009/05/05 14:01:50 bmorin
- SPR1 1539: photTrendCS ignores empty cs side and can work with many plateaus containing many values -please see also PhotDiffCalTasl
- Revision 1.8 2009/02/06 16:04:11 bmorin
- fixed bug
- Revision 1.7 2009/02/04 16:22:51 bmorin
- this task now works with PhotTrendCSProducts object
- Revision 1.6 2008/12/22 11:40:58 bmorin
- SPR 876
- Revision 1.5 2008/12/10 16:38:37 bmorin
- added version

1.222. PhotTrendProducts

Full Name:	herschel.pacs.signal.PhotTrendProducts
Type:	Java Class -
Import:	from herschel.pacs.signal import PhotTrendProducts

History

- 2009-01-27 : (bm) initial version

1.223. photUv2Pq

Full Name:	herschel.pacs.toolboxes.phot.photUv2Pq
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.phot import photUv2Pq

Description

Convert a single pair of bolometer array coordinates u,v to subarray

coordinates p,q and issue appropriate warning if point is outside subarrays or outside array.

API Summary

Jython Syntax
`(p,q) = photUv2Pq(u,v,camera[,model="FM"][,scope="BASE"] [,CalibrationTree calTree])`

Properties
Double **u** [INPUT, MANDATORY, default=NO default value]
Double **v** [INPUT, MANDATORY, default=NO default value]
String **camera** [INPUT, MANDATORY, default=NO default value]
String **model** [INPUT, OPTIONAL, default=default value: "FM"]
String **scope** [INPUT, OPTIONAL, default=default value: "BASE"]
CalibrationTree **calTree** [INPUT, OPTIONAL, default=NO default value]
Double **p** [OUTPUT, OPTIONAL, default=NO default value]
Double **q** [OUTPUT, OPTIONAL, default=NO default value]

Miscellaneous

This jython function is intended for cal analysis procedures, manual sanity checks etc., not for bulk processing which should use direct calfile access

API details

Properties

Double u [INPUT, MANDATORY, default=NO default value]

array coordinate

Double v [INPUT, MANDATORY, default=NO default value]

array coordinate

String camera [INPUT, MANDATORY, default=NO default value]

"blue" or "red"

String model [INPUT, OPTIONAL, default=default value: "FM"]

String scope [INPUT, OPTIONAL, default=default value: "BASE"]

CalibrationTree calTree [INPUT, OPTIONAL, default=NO default value]

if a caltree is supplied, this will override the model and scope settings

Double p [OUTPUT, OPTIONAL, default=NO default value]

subarray coordinate

Double q [OUTPUT, OPTIONAL, default=NO default value]

subarray coordinate For positions that are off the array the string "off" is returned

See also

- ???
- ???

History

- 20060606 DL Creation
- 20070705 DL fixed eval syntax
- 20080305 DL new cal framework
- 20081027 WDM dp-pacs
- 20081125 DL dp-pacs again

1.224. photUv2Yz

Full Name:	herschel.pacs.toolboxes.phot.photUv2Yz
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.phot import photUv2Yz

Description

Convert a single pair of bolometer array coordinates u,v together with a

chopper angle alpha to instrument coordinates y,z. For ILT tests, calfile versions exist in which y,z correspond to y,x of the xy stage. Convert array coordinates of point in the focal plane to instrument coordinates on sky. See also PICC-ME-TN-019

API Summary

Jython Syntax
`(y,z) = photUv2Yz(u,v,alpha,camera[,model="FM"][,scope="BASE"][,CalibrationTree calTree])`

Properties
`Double u [INPUT, MANDATORY, default=NO default value]`
`Double v [INPUT, MANDATORY, default=NO default value]`
`Double alpha [INPUT, MANDATORY, default=NO default value]`
`String camera [INPUT, MANDATORY, default=NO default value]`
`String model [INPUT, OPTIONAL, default=default value: "FM"]`
`String scope [INPUT, OPTIONAL, default=default value: "BASE"]`
`CalibrationTree calTree [INPUT, OPTIONAL, default=NO default value]`
`Double y [OUTPUT, MANDATORY, default=NO default value]`
`Double z [OUTPUT, MANDATORY, default=NO default value]`

Miscellaneous

This jython function is intended for cal analysis procedures, manual sanity checks etc., but not for bulk calibration of PACS data which can be done more efficiently by directly using the calfile

API details

Properties

`Double u [INPUT, MANDATORY, default=NO default value]`

Array coordinate of point [mm]

`Double v [INPUT, MANDATORY, default=NO default value]`

Array coordinate of point [mm]

`Double alpha [INPUT, MANDATORY, default=NO default value]`

Chopper angle [degree]

String camera [INPUT, MANDATORY, default=NO default value]

"blue" or "red"

String model [INPUT, OPTIONAL, default=default value: "FM"]

String scope [INPUT, OPTIONAL, default=default value: "BASE"]

CalibrationTree calTree [INPUT, OPTIONAL, default=NO default value]

if a caltree is supplied, this will override the model and scope settings

Double y [OUTPUT, MANDATORY, default=NO default value]

instrument coordinate

Double z [OUTPUT, MANDATORY, default=NO default value]

instrument coordinate

See also

- ???
- ???

History

- 05/07/2006 DL Creation
- 05/03/2008 DL new cal framework

1.225. photYz2Uv

Full Name:	herschel.pacs.toolboxes.phot.photYz2Uv
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.phot import photYz2Uv

Description

Convert a single pair of bolometer instrument coordinates y,z together with a

chopper angle alpha to array coordinates u,v. For ILT tests, calfile versions exist in which y,z correspond to y,x of the xy stage. Convert instrument coordinates on sky to array coordinates of a point in the focal plane to instrument. See also PICC-ME-TN-019

API Summary

Jython Syntax
`(u,v) = photYz2Uv(y,z,alpha,[,model="FM"][,scope="BASE"] [,CalibrationTree calTree])`

Properties
`Double y [INPUT, MANDATORY, default=NO default value]`
`Double z [INPUT, MANDATORY, default=NO default value]`
`Double alpha [INPUT, MANDATORY, default=NO default value]`
`String camera [INPUT, MANDATORY, default=NO default value]`
`String model [INPUT, OPTIONAL, default=default value: "FM"]`
`String scope [INPUT, OPTIONAL, default=default value: "BASE"]`
`CalibrationTree calTree [INPUT, OPTIONAL, default=NO default value]`
`Double u [OUTPUT, MANDATORY, default=NO default value]`
`Double v [OUTPUT, MANDATORY, default=NO default value]`

Miscellaneous

This jython function is intended for cal analysis procedures, manual sanity checks etc., but not for bulk calibration of PACS data which can be done more efficiently by directly using the calfile

API details

Properties

`Double y [INPUT, MANDATORY, default=NO default value]`

Instrument coordinate [arcsec, mm for ILT calfile version]

`Double z [INPUT, MANDATORY, default=NO default value]`

Instrument coordinate [arcsec, mm for ILT calfile version]

`Double alpha [INPUT, MANDATORY, default=NO default value]`

Chopper angle [degree]

String camera [INPUT, MANDATORY, default=NO default value]

"blue" or "red"

String model [INPUT, OPTIONAL, default=default value: "FM"]

String scope [INPUT, OPTIONAL, default=default value: "BASE"]

CalibrationTree calTree [INPUT, OPTIONAL, default=NO default value]

if a caltree is supplied, this will override the model and scope settings

Double u [OUTPUT, MANDATORY, default=NO default value]

Array coordinate [mm]

Double v [OUTPUT, MANDATORY, default=NO default value]

Array coordinate [mm]

See also

- ???
- ???

History

- 2006-07-10 - DL: Creation
- 2008-03-05 - DL: update to new cal framework

1.226. PlotCubeWaveFluxTask

Full Name:	herschel.pacs.toolboxes.common.PlotCubeWaveFluxTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import PlotCubeWaveFluxTask

Description

function that plots the PacsCube flux over wavelengths for specified resetintervals and

spatial pixels. Overplots 16 spectral pixels.

API Summary

Jython Syntax
`Void plotCubeWaveFlux(PacsCube cube, int xpix, int ypix, int resets )`

Properties
`PacsCube cube [INPUT, MANDATORY, default=NO default value]`
`int xpix [INPUT, MANDATORY, default=NO default value]`
`int ypix [INPUT, MANDATORY, default=NO default value]`
`int resets [INPUT, MANDATORY, default=NO default value]`

API details

Properties

PacsCube cube [INPUT, MANDATORY, default=NO default value]
input PacsCube

int xpix [INPUT, MANDATORY, default=NO default value]
x position of desired spatial pixel signals (0-4)

int ypix [INPUT, MANDATORY, default=NO default value]
y position of desired spatial pixel signals (0-4)

int resets [INPUT, MANDATORY, default=NO default value]
reset interval length for the layers

History

- 20081124 JS 1.0 initial version

1.227. PlotDmcMaskRampsTask

Full Name:	herschel.pacs.toolboxes.common.PlotDmcMaskRampsTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import PlotDmcMaskRampsTask

Description

function that overplots mask and dmc header data vs time for (averaged Ramps)

API Summary

Jython Syntax
`Void plotDmcMaskRamps(Ramps ramp, int xpix, int ypix, String mask, String dmc)`

Properties
`Ramps ramp [INPUT, MANDATORY, default=NO default value]`
`int xpix [INPUT, MANDATORY, default=NO default value]`
`int ypix [INPUT, MANDATORY, default=NO default value]`
`String mask [INPUT, MANDATORY, default=NO default value]`
`String dmc [INPUT, MANDATORY, default=NO default value]`

API details

Properties

Ramps ramp [INPUT, MANDATORY, default=NO default value]
input ramp

int xpix [INPUT, MANDATORY, default=NO default value]
x position of desired pixel signals

int ypix [INPUT, MANDATORY, default=NO default value]
y position of desired pixel signals

String mask [INPUT, MANDATORY, default=NO default value]
mask name

String dmc [INPUT, MANDATORY, default=NO default value]
Dmc header parameter name

History

- 20081120 JS 1.0 initial version

1.228. PlotHkDbTask

Full Name:	herschel.pacs.toolboxes.common.PlotHkDbTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import PlotHkDbTask

Description

Visualization tool for HK parameter from the Database and also in Near Real Time.

API Summary

Jython Syntax
`hkTable = plotHkDb(hkParameter [,startTime=startTime, stopTime=stopTime] [asRaw=asRaw] [stepSize=stepSize])`

Properties
`String` **`hkParameter`** `[INPUT, MANDATORY, default=NO default value]`
`TableDataset` **`HKTable`** `[OUTPUT, MANDATORY, default=NO default value]`
`FineTime` **`startTime/stopTime`** `[INPUT, OPTIONAL, default=NO default value]`
`Long` **`asRaw`** `[INPUT, OPTIONAL, default=default asRaw=0 which means converted parameter values]`
`Long` **`stepSize`** `[INPUT, OPTIONAL, default=default stepSize=10 which means]`

API details

Properties

`String hkParameter [INPUT, MANDATORY, default=NO default value]`

String array containing the selected HK Parameter

`TableDataset HKTable [OUTPUT, MANDATORY, default=NO default value]`

TableDataset containing the selected HK parameter inclusive time (currently 0.0 is given if a column is not filled !

`FineTime startTime/stopTime [INPUT, OPTIONAL, default=NO default value]`

Starttime / Stoptime interval to use

`Long asRaw [INPUT, OPTIONAL, default=default asRaw=0 which means converted parameter values]`

True when raw parameter shall be extracted.

`Long stepSize [INPUT, OPTIONAL, default=default stepSize=10 which means]`

that every 10 new parameter the plot is updated Update Plot after stepSize parameter
True when raw parameter shall be extracted. Examples : --------- # A TimeSelector

Long stepSize [INPUT, OPTIONAL, default=default stepSize=10 which means]
GUI permits convenient selection of a time interval, update every 100 values hktab = plotHkDb("HD_APID",stepSize=100) # Access the data in between the given time interval (Wed May 10 10:22:33 CEST 2006,Thu Sep 28 10:22:33 CEST 2006) from java.util import Date hktab = plotHkDb(("HD_APID", FineTime(Date(106,4,10,10,22,33)), FineTime(Date(106,8,28,10,22,33)))

History

- 1.0 16-Jun-2006 EkW The first version is a prototype.
- TODO : Does not yet work for String parameter !
- TODO : Empty values set to 0.0 in the moment

1.229. PlotHkNrtTask

Full Name:	herschel.pacs.toolboxes.common.PlotHkNrtTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import PlotHkNrtTask

Description

Visualization tool for HK parameter in NRT from the Router

API Summary

Jython Syntax
`plotHkNrt(hkParameter [asRaw=asRaw] [,routerHost= routerHost] [routerPort=routerPort] [stepSize=stepSize] [range = range))`

Properties
`String` **`hkParameter`** `[INPUT, MANDATORY, default=NO default value]`
`Long` **`asRaw`** `[INPUT, OPTIONAL, default=default asRaw=0 which means converted parameter values]`
`Long` **`routerhost`** `[INPUT, OPTIONAL, default=default routerHost=irsun01]`
`Long` **`routerport`** `[INPUT, OPTIONAL, default=default routerPort= 9877]`
`Long` **`stepSize`** `[INPUT, OPTIONAL, default=default stepSize=10 which means]`
`Integer` **`range`** `[INPUT, OPTIONAL, default=default range=1000]`

API details

Properties

`String hkParameter [INPUT, MANDATORY, default=NO default value]`

String array containing the selected HK Parameter

`Long asRaw [INPUT, OPTIONAL, default=default asRaw=0 which means converted parameter values]`

True when raw parameter shall be extracted.

`Long routerhost [INPUT, OPTIONAL, default=default routerHost=irsun01]`

The Router Host

`Long routerport [INPUT, OPTIONAL, default=default routerPort= 9877]`

The Router port

Long stepSize [INPUT, OPTIONAL, default=default stepSize=10 which means]

that every 10 new parameter the plot is updated Update Plot after stepSize parameter

Integer range [INPUT, OPTIONAL, default=default range=1000]

Range of the Plot Window Examples : --------- # A TimeSelector GUI
permits convenient selection of a time interval, update every 10 values, us
port 9888 plotHkNrt("HD_APID",routerPort= 9888, routerHost="irsun01")
plotHkNrt("HD_APID",range=2000)

History

- 1.0 16-Jun-2006 EkW The first version is a prototype
- 1.1 05-Sep-2006 EkW add range parameter

1.230. PlotSignalDmcMaskTask

Full Name:	herschel.pacs.toolboxes.common.PlotSignalDmcMaskTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import PlotSignalDmcMaskTask

Description

function that overplots ask and dmc header data vs time for (averaged Ramps)

API Summary

Jython Syntax
`Void plotSignalDmcMask(Frames frame, int xpix, int ypix [, String dmc][, String mask])`

Properties
`Frames frame [INPUT, MANDATORY, default=NO default value]`
`int xpix [INPUT, MANDATORY, default=NO default value]`
`int ypix [INPUT, MANDATORY, default=NO default value]`
`String dmc [INPUT, OPTIONAL, default=default value: dmc]`
`String mask [INPUT, OPTIONAL, default=default value: mask]`

API details

Properties

Frames frame [INPUT, MANDATORY, default=NO default value]
input ramp

int xpix [INPUT, MANDATORY, default=NO default value]
x position of desired pixel signals

int ypix [INPUT, MANDATORY, default=NO default value]
y position of desired pixel signals

String dmc [INPUT, OPTIONAL, default=default value: dmc]
Dmc header parameter name

String mask [INPUT, OPTIONAL, default=default value: mask]
mask name

History

- 20081121 JS 1.0 initial version

1.231. plotSignalHK

Full Name:	herschel.pacs.toolboxes.common.plotSignalHK
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import plotSignalHK

Description

function that overplots signals and HK data vs time

API Summary

Jython Syntax
`Void plotSignalHK(String filename, int xpix, int ypix, String HkType, String HkParameter)`

Properties
`String filename [INPUT, MANDATORY, default=NO default value]`
`int xpix [INPUT, MANDATORY, default=NO default value]`
`int ypix [INPUT, MANDATORY, default=NO default value]`
`String filename [INPUT, MANDATORY, default=NO default value]`
`String HkType [INPUT, MANDATORY, default=No default value]`
`String HkParameter [INPUT, MANDATORY, default=NO default value]`

API details

Properties

`String filename [INPUT, MANDATORY, default=NO default value]`

input tm file name

`int xpix [INPUT, MANDATORY, default=NO default value]`

x position of desired pixel signals

`int ypix [INPUT, MANDATORY, default=NO default value]`

y position of desired pixel signals

`String filename [INPUT, MANDATORY, default=NO default value]`

input tm file name

`String HkType [INPUT, MANDATORY, default=No default value]`

HK type (e.g. "ESSENTIAL_HK"), if it is "DMC" DEC/MEC HK is read

`String HkParameter [INPUT, MANDATORY, default=NO default value]`

HK Parameter

History

- 20051207 JS 1.0 initial version

1.232. pointSourceL1

Full Name:	herschel.pacs.spg.pipeline.ipipe.phot.pointSourceL1
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.ipipe.phot import pointSourceL1

History

- 1.0 20090313 EkW
- 2.0 20100120 EkW Prepare for Operation

1.233. pointSourceL2

Full Name:	herschel.pacs.spg.pipeline.ipipe.phot.pointSourceL2
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.ipipe.phot import pointSourceL2

History

- 1.0 20090313 EkW

1.234. polynomialDerivation

Full Name:	herschel.pacs.toolboxes.numeric.polynomialDerivation
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import polynomialDerivation

Description

Jython function that determines the polynomial coefficients of the first derivation of a polynomial

API Summary

Jython Syntax
`Double deriv = polynomialDerivation(Double1d coefficients)`

Properties
`Double1d coefficients [INPUT, MANDATORY, default=NO default value]`
`Double parameter [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Double1d coefficients [INPUT, MANDATORY, default=NO default value]`
coefficients of the polynomial to derive

`Double parameter [OUTPUT, MANDATORY, default=NO default value]`
result coefficient array of derivated polynomial

History

- 1.0 20060505 JS initial version
- 1.1 20060512 JS Double1d import inside function

1.235. PopulatePacsPoolFromDatabaseTask

Full Name:	herschel.pacs.toolboxes.common.PopulatePacsPoolFromDatabaseTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import PopulatePacsPoolFromDatabaseTask

Description

unknown

Example

Example 1: :

```
c  = populatePacsPoolFromDatabase(32323232l, -"pacs_fm_ilt_3", -"TestDatabase")
```

API Summary

Jython Syntax

```
populatePacsPool ( [,obsid] [database] [, poolName || [,
startTime, stopTime] [, hklimit] [, sclimit] )
```

Properties

`Long obsid [INPUT, OPTIONAL, default=default : 0]`
`String database [INPUT, OPTIONAL, default=default : "EkkiDummy"]`
`String poolName [INPUT, OPTIONAL, default=default : Test1]`
`FineTime startTime and stopTime [INPUT, OPTIONAL, default=default : FineTime(0)]`
`Integer hklimit [INPUT, OPTIONAL, default=default : 1000]`
`Integer sclimit [INPUT, OPTIONAL, default=default : 100]`

API details

Properties

Long obsid [INPUT, OPTIONAL, default=default : 0]

OBSID for the parameter to use (overwrites the startTime and stopTime parameter)

String database [INPUT, OPTIONAL, default=default : "EkkiDummy"]

Database from where to get the Telemetry

String poolName [INPUT, OPTIONAL, default=default : Test1]

Name of the generated Pool

FineTime startTime and stopTime [INPUT, OPTIONAL, default=default : FineTime(0)]

Starttime and stoptime interval to use

Integer hklimit [INPUT, OPTIONAL, default=default : 1000]
Number of HK TmSourcePackets used for a Pool enty

Integer sclimit [INPUT, OPTIONAL, default=default : 100]
Number of Science SPUBuffers used for a Pool enty

History

- 1.0 08-JApr-2008 EkW

1.236. PopulatePacsPoolFromFilesTask

Full Name:	herschel.pacs.toolboxes.common.PopulatePacsPoolFromFilesTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import PopulatePacsPoolFromFilesTask

Description

unknown

Example

Example 1: :

```
c = populatePacsPoolFromFiles(f, 111, -"Level0Test1")
c = populatePacsPoolFromFiles(poolName =  -"Level0Test1", startDirectory="..")
```

API Summary

Jython Syntax

```
populatePacsPoolFromFiles ([, filename] [, obsid] [, poolName]
[,startDirectory] [, pattern] [, walk] [, hklimit] [, sclimit] )
```

Properties

String **filename** [INPUT, OPTIONAL, default=default : "EkkiDummy"]
String **obsid** [INPUT, OPTIONAL, default=default : -1]
String **poolName** [INPUT, OPTIONAL, default=default : Test1]
String **startDirectory** [INPUT, OPTIONAL, default=default : "EkkiDummy"]
String **pattern** [INPUT, OPTIONAL, default=default : "*.tm"]
Boolean **walk** [INPUT, OPTIONAL, default=default : False]
Integer **hklimit** [INPUT, OPTIONAL, default=default : 10000]
Integer **sclimit** [INPUT, OPTIONAL, default=default : 10000]
String **store** [INPUT, OPTIONAL, default=no default value]

API details

Properties

String filename [INPUT, OPTIONAL, default=default : "EkkiDummy"]

Filename

String obsid [INPUT, OPTIONAL, default=default : -1]

OBSID will not be extracted, all data saved under this obsid

String poolName [INPUT, OPTIONAL, default=default : Test1]

Name of the Pool

String startDirectory [INPUT, OPTIONAL, default=default : "EkkiDummy"]

Start directory for pattern search

String pattern [INPUT, OPTIONAL, default=default : "*.tm"]

File pattern to search for

Boolean walk [INPUT, OPTIONAL, default=default : False]

If walk is true, subdirectories will be considered

Integer hklimit [INPUT, OPTIONAL, default=default : 10000]

Number of HK TmSourcePackets used for a Pool enty

Integer sclimit [INPUT, OPTIONAL, default=default : 10000]

Number of Science SPUBuffers used for a Pool enty

String store [INPUT, OPTIONAL, default=no default value]

Store containing the PointingProduct

History

- 1.0 09-Jan-2008 EkW subset from PopulatePacsPoolTask
- 2.0 14-Oct-2008 EkW operational version

1.237. PopulatePacsPoolTask

Full Name:	herschel.pacs.toolboxes.common.PopulatePacsPoolTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import PopulatePacsPoolTask

Description

unknown

API Summary

Jython Syntax
`populatePacsPool ([startDirectory] [, pattern] [, noWalk] [, filename] [, database] [, obsid] [, startTime] [, stopTime] [, listen] [, poolName] [,poolType] [, hklimit] [, sclimit] [, noObsid] [, asRaw])`

Properties
`String` **`obsid`** `[INPUT, OPTIONAL, default=default : "EkkiDummy"]`
`String` **`pattern`** `[INPUT, OPTIONAL, default=default : "*.tm"]`
`Boolean` **`noWalk`** `[INPUT, OPTIONAL, default=default : True]`
`String` **`filename`** `[INPUT, OPTIONAL, default=default : "EkkiDummy"]`
`String` **`database`** `[INPUT, OPTIONAL, default=default : "EkkiDummy"]`
`Long` **`obsid`** `[INPUT, OPTIONAL, default=default : 0]`
`FineTime` **`startTime and stopTime`** `[INPUT, OPTIONAL, default=default : FineTime(0)]`
`Boolean` **`listen`** `[INPUT, OPTIONAL, default=default : False]`
`String` **`poolname`** `[INPUT, OPTIONAL, default=default : Test1]`
`String` **`poolname`** `[INPUT, OPTIONAL, default=default : DBPool]`
`String` **`seq`** `[INPUT, OPTIONAL, default=no default value]`
`Integer` **`hklimit`** `[INPUT, OPTIONAL, default=default : 1000]`
`Integer` **`sclimit`** `[INPUT, OPTIONAL, default=default : 100]`
`Boolean` **`asRaw`** `[INPUT, OPTIONAL, default=default asRaw=0 which means converted parameter values]`

API details

Properties

`String` obsid `[INPUT, OPTIONAL, default=default : "EkkiDummy"]`
In case we start from files it is the start directory where to find the files

`String` pattern `[INPUT, OPTIONAL, default=default : "*.tm"]`
In case we start from files it is the file pattern to search for

Boolean noWalk [INPUT, OPTIONAL, default=default : True]

In case we start from filenames and directory, False means we are also walking through sub-directories

String filename [INPUT, OPTIONAL, default=default : "EkkiDummy"]

In case we start from one file it is the filename

String database [INPUT, OPTIONAL, default=default : "EkkiDummy"]

Database from where to get the Telemetry

Long obsid [INPUT, OPTIONAL, default=default : 0]

OBSID for the parameter to use (overwrites the startTime and endTime parameter)

FineTime startTime and stopTime [INPUT, OPTIONAL, default=default : FineTime(0)]

Starttime and stoptime interval to use

Boolean listen [INPUT, OPTIONAL, default=default : False]

Listen in the Database for finished Observations

String poolname [INPUT, OPTIONAL, default=default : Test1]

Name of the generated Pool

String poolname [INPUT, OPTIONAL, default=default : DBPool]

Type of the generated Pool : "DBPool" or "LocalStore"

String seq [INPUT, OPTIONAL, default=no default value]

PacketSequence in case we start form it

Integer hklimit [INPUT, OPTIONAL, default=default : 1000]

Number of HK TmSourcePackets used for a Pool enty

Integer sclimit [INPUT, OPTIONAL, default=default : 100]

Number of Science SPUBuffers used for a Pool enty

Boolean asRaw [INPUT, OPTIONAL, default=default asRaw=0 which means converted parameter values]

True when raw parameter shall be extracted. Examples : stat=populatePacsPool(startDirectory=".",pattern="*.tm",noObsid=True, poolName="Bulk1")

History

- 0.1 24-Jan-2007 EkW First draft wit very primitive Interface to file list
- 1.0 19-Mar-2007 EkW Add Database implementation
- 1.1 21-Mar-2007 EkW Add PoolManeger

1.238. prototype_pacs_spec_pipeline

Full Name:	herschel.pacs.toolboxes.spg.prototype_pacs_spec_pipeline
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spg import prototype_pacs_spec_pipeline

History

- 1.0 20071019 JS initial version
- 1.1 20080222 JS adapted according to new cal framework
- 1.2 20080311 JS adapt sequence of modules and add specAssignRaDec
- 1.3 20080327 JdJ reorganized original specipe.py script to simplify handling frames/ramps

1.239. rampDeglitch

Full Name:	herschel.pacs.toolboxes.spec.rampDeglitch
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spec import rampDeglitch

Description

Jython function to deglitch a given ramp

This function reads a Ramps object and searches for glitches in each ramp by computing the mean and the standard deviation of the differential signal between two readouts after rejecting a user given number of first and last readouts of the ramp. Additionally, a user defined number of the lowest and highest values are rejected before computing the statistics. If the deviation from the mean is higher than a user given factor times the standard deviation, the readout is flagged as glitch. This procedure is repeated n_iterate times. The found glitches are corrected by using the mean differential signal. The readouts after a glitch detection are searched for glitches by using a smaller sigma factor. The function returns the corrected Ramps object.

API Summary

Jython Syntax
`Ramps ramp = rampDeglitch(file, pacsramps, n_firstReject, n_lastReject, sigma1, sigma2, min_npix1, min_npix2, n_iterate, n_lowReject, n_highReject)`

Properties
`String` **`fname`** `[INPUT, MANDATORY, default=NO default value]`
`Ramps` **`pacsramps`** `[INPUT, MANDATORY, default=NO default value]`
`int` **`n_firstReject`** `[INPUT, MANDATORY, default=NO default value]`
`int` **`n_lastReject`** `[INPUT, MANDATORY, default=NO default value]`
`Double` **`sigma1`** `[INPUT, MANDATORY, default=NO default value]`
`Double` **`sigma2`** `[INPUT, MANDATORY, default=NO default value]`
`int` **`min_npix1`** `[INPUT, MANDATORY, default=NO default value]`
`int` **`min_npix2`** `[INPUT, MANDATORY, default=NO default value]`
`int` **`n_iterate`** `[INPUT, MANDATORY, default=NO default value]`
`int` **`n_lowReject`** `[INPUT, MANDATORY, default=NO default value]`
`int` **`n_highReject`** `[INPUT, MANDATORY, default=NO default value]`
`Ramps` **`ramp`** `[OUTPUT, MANDATORY, default=NO default value]`
`FILE` **`file`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`String fname [INPUT, MANDATORY, default=NO default value]`
file name of input file

Ramps pacsramps [INPUT, MANDATORY, default=NO default value]
Ramps Object which is to be corrected

int n_firstReject [INPUT, MANDATORY, default=NO default value]
number of first ramp readouts which are not used to compute statistics (5)

int n_lastReject [INPUT, MANDATORY, default=NO default value]
number of last ramp readouts which are not used to compute statistics (3)

Double sigma1 [INPUT, MANDATORY, default=NO default value]
factor of standard deviation to define threshold for glitch detection (5)

Double sigma2 [INPUT, MANDATORY, default=NO default value]
factor of standard deviation to define threshold for secondary glitch detection (1.5)

int min_npix1 [INPUT, MANDATORY, default=NO default value]
minimum number of remaining valid readouts in ramps to detect glitches by using statistics (20)

int min_npix2 [INPUT, MANDATORY, default=NO default value]
minimum number of remaining valid readouts in ramps to detect also secondary glitches (20)

int n_iterate [INPUT, MANDATORY, default=NO default value]
number of iterations to detect primary glitches (4)

int n_lowReject [INPUT, MANDATORY, default=NO default value]
number of lowest values in ramp to be rejected before computing statistics (3)

int n_highReject [INPUT, MANDATORY, default=NO default value]
number of highest values in ramp to be rejected before computing statistics (3)

Ramps ramp [OUTPUT, MANDATORY, default=NO default value]
returned deglitched Ramps object

FILE file [OUTPUT, MANDATORY, default=NO default value]
file pointer of output file An output file is generated which contains the number of found glitches and indicates the filename, the time, the detector id, the voltage step and the readout index of the glitch.

See also

- ISO Handbook Vol IV, PHT, 7.2.9: Ramp deglitching

History

- 1.0 20050510 JS initial version
- 1.1 20050517 JS bug fix in threshold calculation
- 1.2 20050524 JS imports moved inside function

- 1.3 20050621 JS got rid of global variables,
- GLITCH and GLITCHTAIL mask set
- 1.4 20050804 JS header adopted to jtags syntax
- 1.5 20060306 JS getRampValues() -> getSignal()

1.240. rampsL05

Full Name:	herschel.pacs.spg.pipeline.ipipe.spec.rampsL05
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.ipipe.spec import rampsL05

History

- 1.0 20090313 JS initial version
- 1.1 20090624 JS updated
- 1.2 20090723 JS introduce slicing

1.241. ReadAttitudeHistoryTask

Full Name:	herschel.pacs.toolboxes.common.ReadAttitudeHistoryTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import ReadAttitudeHistoryTask

Description

Reads the instantaneous pointing product covering the same time as the dataframes in the pdfs

API Summary

Jython Syntax
`PointingProduct pp = readAttitudeHistory(DataFrameSequence pdfs)`

Properties
`DataFrameSequence` **`pdfs`** `[INPUT, MANDATORY, default=NO default value]`
`PontingProduct` **`pp`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`DataFrameSequence pdfs [INPUT, MANDATORY, default=NO default value]`
A PACS DataFrameSequence

`PontingProduct pp [OUTPUT, MANDATORY, default=NO default value]`
The Pointing Product covering the same time as the dataframes of the input sequence

History

- 0.1 19-Jul-2006 EkW Initial version of this Task basing on script of BV
- - basing on file pool and prototype of the PointingProduct
- - not fo production : for End2End test

1.242. ReadDfDbTask

Full Name:	herschel.pacs.toolboxes.common.ReadDfDbTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import ReadDfDbTask

Description

If this task invoked with an observation id, then gets all Pacs dataframes

associated with that id. If invoked without an observation is, then calls TestExecutionBrowser. Waits for the user to hit "export" in the browser. Gets the exported time array of selected TestExecutions and extracts DataFrames into DataFrameSequence. As of 22 June 2006, similar code invoking the TestExecutionBrowser hangs if run in jide, so, run it in jide_new. As of the first delivery, there is no way to close a browser from within the jide session, so, you can end up with a screen full of browsers that can no longer be used. This problem will be addressed in a later version. Note: This task is a prototype.

API Summary

Jython Syntax
`DataFrameSequence dfSeq = readDfDb([obsid=obsid])`

Property
`DataFrameSequence frame [OUTPUT, MANDATORY, default=NO default value]`

API details

Property

`DataFrameSequence frame [OUTPUT, MANDATORY, default=NO default value]`
returned DataFrameSequence

History

- 1.0 28-Aug-2006 JJ Start w/ ReadTmDbTask for first cut.
- 1.1 11-Nov-2006 JJ Destroy browser return frames for blocks of test execs.

1.243. ReadHkDbTask

Full Name:	herschel.pacs.toolboxes.common.ReadHkDbTask
Type:	Java Task -
Import:	from herschel.pacs.toolboxes.common import ReadHkDbTask

Description

Extract HK parameter from the Database

Example

Example 1: Jython Example

```
IA>> t=readHkDb(["HD_APID","N_DPU_Current"]) <br>
```

API Summary

Jython Syntax

```
TableDataset hkTable = readHkDb(String[] hkParameter [,Long
obsid=obsid][,Boolean asRaw=asRaw] <p>
[,FineTime startTime = startTime ,FineTime stopTime = stopTime]
[,stepSize=stepSize] <p>
[,chunkSize=chunkSize]) <p>
```

Properties

String **hkParameter** [INPUT, MANDATORY, default=NO default value]
Long **obsid** [INPUT, OPTIONAL, default=0]
Boolean **asRaw** [INPUT, OPTIONAL, default=False]
FineTime **startTime** [INPUT, OPTIONAL, default=0]
FineTime **stopTime** [INPUT, OPTIONAL, default=0]

API details

Properties

String hkParameter [INPUT, MANDATORY, default=NO default value]

input parameter String array

Long obsid [INPUT, OPTIONAL, default=0]

optional: OBSID

Boolean asRaw [INPUT, OPTIONAL, default=False]

raw (True) or converted (False) parameter

FineTime startTime [INPUT, OPTIONAL, default=0]

start time as finetime

FineTime stopTime [INPUT, OPTIONAL, default=0]
stop time as finetime

1.244. ReadSimulatorDataTask

Full Name:	herschel.pacs.toolboxes.common.ReadSimulatorDataTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import ReadSimulatorDataTask
Category:	pacs/toolboxes/common

Description

Read Simulator FITS files and import the data into Frames class

Example

Example 1: flag bad pixel

```
frames = readSimulatorData("Simfile.fits", -"PHOTBLUE")
```

API Summary

Jython Syntax

```
frames = readSimulatorData(filename, type , calTree=calTree,
chopperSkyAngle=chopperSkyAngle , chopperAngle=chopperAngle ,
chopperAngleRedundant=chopperAngleRedundant,
arrayInstrument=arrayInstrument, subArrayArray=subArrayArray)
```

Properties
filename **frames** [INPUT, MANDATORY, default=no default value]
type **frames** [INPUT, MANDATORY, default=no default value]
PacsCal **calTree** [INPUT, OPTIONAL, default=default value :null]
ChopperSkyAngle **chopperSkyAngle** [INPUT, OPTIONAL, default=default value : null]
ChopperAngle **chopperAngle** [INPUT, OPTIONAL, default=default value : null]
ChopperAngleRedundant **chopperAngleRedundant** [INPUT, OPTIONAL, default=default value : null]
ArrayInstrument **arrayInstrument** [INPUT, OPTIONAL, default=default value : null]
SubArrayArray **subArrayArray** [INPUT, OPTIONAL, default=default value : null]
Frames **frames** [OUTPUT, MANDATORY, default=no default value]

Limitations

No Limitation

API details

Properties

filename frames [INPUT, MANDATORY, default=no default value]

String filename

type frames [INPUT, MANDATORY, default=no default value]
String type : PHOTBLUE, PHOTRED, SPECBLUE, SPECRED

PacsCal calTree [INPUT, OPTIONAL, default=default value :null]
optional Calibration Tree

ChopperSkyAngle chopperSkyAngle [INPUT, OPTIONAL, default=default value : null]
optional ChopperSkyAngle

ChopperAngle chopperAngle [INPUT, OPTIONAL, default=default value : null]
optional ChopperAngle

ChopperAngleRedundant chopperAngleRedundant [INPUT, OPTIONAL, default=default value : null]
optional ChopperAngleRedundant

ArrayInstrument arrayInstrument [INPUT, OPTIONAL, default=default value : null]
optional ArrayInstrument

SubArrayArray subArrayArray [INPUT, OPTIONAL, default=default value : null]
optional SubArrayArray

Frames frames [OUTPUT, MANDATORY, default=no default value]
Frames

See also

- ???

History

- 1.0 18-May-2007 first draft
- 1.1 20-Nov-2007 Rene pixel ra and dec handling modified
- 1.2 19-Dec-2007 EkW correct Ra Dec assignment
- 1.3 20-Dec-2007 EkW Introduce RA and DEC ins status again -> needed for MadMap
- 1.4 22-Feb-2008 EkW Adopt to new calibration framework

1.245. ReadTmDbTask

Full Name:	herschel.pacs.spg.ReadTmDbTask
Type:	Java Task -
Import:	from herschel.pacs.spg import ReadTmDbTask

API Summary

Jython Syntax
`seq = readTmDb([obsid=obsid] [,startTime=startTime, endTime=endTime])`

Properties
`type` **`long obsid OBSID for the paremeter to use (overwrites the startTime and endTime parameter)`** `[INPUT, MANDATORY, default=no default value]`
`type` **`FineTime startTime and endTime Starttime and stoptime interval to use`** `[INPUT, MANDATORY, default=no default value]`
`type` **`Finetime endTime stop time (always together with startTime)`** `[INPUT, MANDATORY, default=no default value]`

API details

Properties

`type long obsid OBSID for the paremeter to use (overwrites the startTime and endTime parameter) [INPUT, MANDATORY, default=no default value]`

`type FineTime startTime and endTime Starttime and stoptime interval to use [INPUT, MANDATORY, default=no default value]`

`type Finetime endTime stop time (always together with startTime) [INPUT, MANDATORY, default=no default value]`

History

- 3.0 07-May-2008 EkW basing on toolboxes/spg/ReafTmDbTask.py

1.246. ReadTmDbTask

Full Name:	herschel.pacs.toolboxes.common.ReadTmDbTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import ReadTmDbTask

Description

Calls TestExecutionBrowser. Waits for the user to hit "export" in the browser.

Get's the exported time array of selected TestExecutions and extracts TmSourcePackets into PacktSequence. As of 22 June 2006, hangs if run in jide, so, run it in jide_new. If OBSID is given it just extract the data for the given observation. Nevertheless the Browser is currently popping up. This will be corrected in further versions.

API Summary

Jython Syntax
`seq = readTmDb([obsid] || [startTime,endTime] )`

Properties
`long obsid [INPUT, Optional, default=NO default value]`
`FineTime startTime [INPUT, Optional, default=default value: 0]`
`FineTime endTime [INPUT, Optional, default=default value: 0]`

API details

Properties

long obsid [INPUT, Optional, default=NO default value]
OBSID

FineTime startTime [INPUT, Optional, default=default value: 0]
start Time

FineTime endTime [INPUT, Optional, default=default value: 0]
end Time

History

- 1.0 16-Jun-2006 JJ The first version is a prototype.
- 1.1 25-Jul-2006 EkW First implemetation, add the obsid option
- 1.1 11-Nov-2006 JJ Destroy browser and return tm for blocks of test execs.
- 2.0 10-Apr-2007 EkW React on access change when using stream

1.247. Reconstruction

Full Name:	herschel.pacs.spg.phot.deglitching.WTMML.reconstruction.Reconstruction
Type:	Java Class -
Import:	from herschel.pacs.spg.phot.deglitching.WTMML.reconstruction import Reconstruction
Category:	class

History

- 28/04/2006 : first java version (ancient classes now removed)
- 28 february 2007 : new java version (new class)

1.248. resampleOnTime

Full Name:	herschel.pacs.toolboxes.common.resampleOnTime
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import resampleOnTime

Description

Jython function that accepts two absolute time arrays of different/same length from

two different measures arrays. The first measure array is then resampled to the length of the second array by using linear inter/extra-polation. Measure1 should be rebinned on timeline2

API Summary

Jython Syntax
`measure1_out = resampleOnTime(Double1d measure1, Double1d timeline1, Double1d timeline2)`

Properties
`Double1d measure1 [INPUT, MANDATORY, default=NO default value]`
`Double1d timeline1 [INPUT, MANDATORY, default=NO default value]`
`Double1d timeline2 [INPUT, MANDATORY, default=NO default value]`
`Double1d measure1_out [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

Double1d measure1 [INPUT, MANDATORY, default=NO default value]
measures array which is to be resampled

Double1d timeline1 [INPUT, MANDATORY, default=NO default value]
timeline array, must have same length as measure1 given in seconds since ....

Double1d timeline2 [INPUT, MANDATORY, default=NO default value]
timeline array, must have same length as measure2

Double1d measure1_out [OUTPUT, MANDATORY, default=NO default value]
resampled array

History

- 1.0 20060711 JS initial version
- 1.1 20060928 JS linear inter/extrapolation used

1.249. RsrfCalTask

Full Name:	herschel.pacs.spg.spec.RsrfCalTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.spec import RsrfCalTask

Description

Task to divide by the Relative Spectral Response Function

The response of a pixel in the PACS spectrometer depends on the wavelength seen by the pixel. This wavelength dependence response is tabulated in the Relative Spectral Response (RSRF) calibration files. In the pacs calibration tree the RSRFs can be found here:

- cal.spectrometer.RsrfB3A
- cal.spectrometer.RsrfB2A
- cal.spectrometer.RsrfB2B
- cal.spectrometer.RsrfR1

This task loops over every band and over every pixels. For every frame, the applicable relative spectral response is determined by interpolating the corresponding RSRF at the wavelength seen by the pixel in that frame. The signal is then divided by that response.

The statistical error is propagated as follows:

```
error = SQRT((error/response)^2+(signal*rsrfError/response)^2)
```

with *error* the statistical error on the detector signal (after/before applying this task), *response* the relative spectral response as interpolated from the RSRF, *signal* the detector signal before applying this task and *rsrfError* the interpolated statistical error in the RSRF calibration table.

API Summary

Jython Syntax

`Frames outFrames = rsrfCal(Frames inFrames[,calTree=calTree]`
`[, rsrfR1=rsrfR1] [, rsrfB2B=rsrfB2B] [, rsrfB2A=rsrfB2A]`
`[, rsrfB3A=rsrfB3A]`
`[, int normalise=1] [, int copy = <copy>])`

Properties
`Frames` **`inFrames`** `[INPUT, MANDATORY, default=NO default value]`
`Frames` **`outFrames`** `[OUTPUT, MANDATORY, default=NO default value]`
`PacsCal` **`calTree`** `[INPUT, OPTIONAL, default=None]`
`RsrfR1` **`rsrfR1`** `[INPUT, OPTIONAL, default=None]`
`RsrfB2B` **`rsrfB2B`** `[INPUT, OPTIONAL, default=None]`
`RsrfB2A` **`rsrfB2A`** `[INPUT, OPTIONAL, default=None]`
`RsrfB3A` **`rsrfB3A`** `[INPUT, OPTIONAL, default=None]`
`Integer` **`normalise`** `[INPUT, Optional, default=default value: 1]`

Properties
Integer **copy** [INPUT, Optional, default=default value: 0]

API details

Properties

Frames inFrames [INPUT, MANDATORY, default=NO default value]

Frames input class for wavelength calculation

Frames outFrames [OUTPUT, MANDATORY, default=NO default value]

Depending on copy keyword a copy of the Frames class with division by Rsrf

PacsCal calTree [INPUT, OPTIONAL, default=None]

cal file access

RsrfR1 rsrfR1 [INPUT, OPTIONAL, default=None]

Force the use of this Calibration file for band R1

RsrfB2B rsrfB2B [INPUT, OPTIONAL, default=None]

Force the use of this Calibration file for band B2B

RsrfB2A rsrfB2A [INPUT, OPTIONAL, default=None]

Force the use of this Calibration file for band B2A

RsrfB3A rsrfB3A [INPUT, OPTIONAL, default=None]

Force the use of this Calibration file for band B3A

Integer normalise [INPUT, Optional, default=default value: 1]

Historical parameter: the RSRF tables are normalised to 1 at the key wavelength of the spectral band. This parameter is ignored.

Integer copy [INPUT, Optional, default=default value: 0]

indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

History

- 1.0 25-Mar-2008 BV Initial version
- 1.1 13-July-2009 JS error propagation introduced
- 1.2 16-Nov-2009 BV Improved URM documentation

1.250. RunMadMap

Full Name:	herschel.pacs.spg.phot.RunMadMap
Type:	Java Class -
Import:	from herschel.pacs.spg.phot import RunMadMap

Description

RunMadMap

This class produces either a Naive map or a Madmap given a tod byte array.

API Summary

Method
SimpleImage **makeProduct**(PacsTodProduct tod, PacsCal calTree, Double maxerror, Integer maxiterations, Boolean runNaiveMapper, Boolean useAvgInvntt)
makeProduct

API details

Method

SimpleImage makeProduct(PacsTodProduct tod, PacsCal calTree, Double maxerror, Integer maxiterations, Boolean runNaiveMapper, Boolean useAvgInvntt)

makeProduct

Given a MADmap TOD product and a calibration tree which includes the Invntt information, run MadMap to produce a SimpleImage.

Arguments

PacsTodProduct **tod** [INPUT, MANDATORY, default=no default value]

TOD Product

PacsCal **calTree** [INPUT, MANDATORY, default=no default value]

Pacs calibration tree

Double **maxerror** [INPUT, OPTIONAL, default=1.E-5]

Madmap max error

Integer **maxiterations** [INPUT, OPTIONAL, default=200]

Madmap max iterations

Boolean **runNaiveMapper** [INPUT, OPTIONAL, default=false]

true = run NaiveMapper, false = run MadMapper

Boolean **useAvgInvntt** [INPUT, OPTIONAL, default=false]

false = use per-detector Invntt data, true = use averaged Invntt data

Return

SimpleImage

SimpleImage makeProduct(PacsTodProduct tod, PacsCal calTree, Double maxerror, Integer maxiterations, Boolean runNaiveMapper, Boolean useAvgInvntt)

Returns either a Naivemap or Madmap with their associated error map and coverage map if any

Errors

`NoSuchFieldException`

fields missing in PacsTodProduct

`HeaderCardException`

error occured in getting nom.tam.fits.Header

`GeneralSecurityException`

error occured in PacsCal tree

`InterruptedException`

error occured while running either NaiveMapper or MadMapper

`IOException`

error occured while running either NaiveMapper or MadMapper

History

- `2008-01-28 - JJ`: First implementation, v0.5
- `2008-02-13 - CL`: Added MadMapper's IOException handling
- `2008-02-20 - CL`: Add temporary files clean up before and after running MadMapper, v1.1
- `2008-02-26 - CL`: Add two Datasets to PacsTodProduct, v1.18
- `2008-03-18 - CL`: Adhere to new MadMapper interface, v1.21
- `2008-03-26 - CL`: Change array orientation to my own understanding, v1.22
- `2008-04-03 - CL`: Add Wcs to final image output, v1.23
- `2008-05-12 - CL`: Add place holder for dealing with different wavelength bands, v1.24
- `2008-05-21 - CL`: Support 3 bands: BL, BS, RED, v1.25
- `2008-06-13 - CL`: Remove saving final image into PacsTodProduct, v1.27
- `2008-07-24 - CL`: Move cleaning up to MakeTodArray.java, v1.28
- `2008-09-25 - CL`: Add logic to run NaiveMapper, v1.30
- `2008-10-31 - CL`: Chunking by scan legs using OnTarget flag in Status, v1.32

1.251. RunMadMapTask

Full Name:	herschel.pacs.spg.phot.RunMadMapTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import RunMadMapTask

Description

RunMadMapTask

This class is the task that calls runMADmap(). It takes a PacsTodProduct, gets the INVNTT calibration file information and runs MADmap,

Example

Example 1: Description runMadMap

```
map = runMadMap(todprod, calTree, filterlength, maxerror, maxiterations,
runNaiveMapper)
map = runMadMap(todprod, calTree)
map = runMadMap(todprod, calTree, runNaiveMapper=Boolean.TRUE)  -//for NaiveMap
```

API Summary

Properties
PacsTodProduct **todproduct** [INPUT, MANDATORY, default=no default value]
PacsCal **calTree** [INPUT, MANDATORY, default=no default value]
Integer **filterlength** [INPUT, OPTIONAL, default=0]
Integer **maxiterations** [INPUT, OPTIONAL, default=200]
Boolean **runNaiveMapper** [INPUT, OPTIONAL, default=false]
SimpleImage **mapimage** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

PacsTodProduct todproduct [INPUT, MANDATORY, default=no default value]

TOD Product

PacsCal calTree [INPUT, MANDATORY, default=no default value]

Pacs calibration tree

Integer filterlength [INPUT, OPTIONAL, default=0]

Madmap max error

Integer maxiterations [INPUT, OPTIONAL, default=200]

Madmap max iterations

Boolean runNaiveMapper [INPUT, OPTIONAL, default=false]
true = run NaiveMapper, false = run MadMapper

SimpleImage mapimage [OUTPUT, MANDATORY, default=no default value]
Contains either a Naivemap or Madmap with their associated error map and coverage map if any

1.252. runPhotometerPointSource_pipelinetest

Full Name:	herschel.pacs.spg.pipeline.oldstyle.runPhotometerPointSource_pipelinetest
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import runPhotometerPointSource_pipelinetest

History

- 1.0 20080417 EkW initial version
- 1.1 20080715 EkW usage of calTree
- 1.2 20080902 EkW use PointingProduct

1.253. runPhotometerPointSource

Full Name:	herschel.pacs.spg.pipeline.oldstyle.runPhotometerPointSource
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import runPhotometerPointSource

History

- 1.0 20080417 EkW initial version
- 1.1 20080715 EkW usage of calTree
- 1.2 20080902 EkW use PointingProduct

1.254. runPhotometerScanMap_pipelinetest

Full Name:	herschel.pacs.spg.pipeline.oldstyle.runPhotometerScanMap_pipelinetest
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import runPhotometerScanMap_pipelinetest

History

- 1.0 20080424 EkW initial version
- 1.1 20080715 EkW usage of calTree
- 1.2 20080902 EkW use PointingProduct

1.255. runPhotometerScanMap

Full Name:	herschel.pacs.spg.pipeline.oldstyle.runPhotometerScanMap
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import runPhotometerScanMap

History

- 1.0 20080424 EkW initial version
- 1.1 20080715 EkW usage of calTree
- 1.2 20080902 EkW use PointingProduct

1.256. runPhotometerScanMapSimple_pipelinetest

Full Name:	herschel.pacs.spg.pipeline.oldstyle.runPhotometerScanMapSimple_pipelinetest
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import runPhotometerScanMapSimple_pipelinetest

History

- 1.0 20080424 EkW initial version
- 1.2 20080902 EkW use PointingProduct

1.257. runPhotometerScanMapSimple

Full Name:	herschel.pacs.spg.pipeline.oldstyle.runPhotometerScanMapSimple
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import runPhotometerScanMapSimple

History

- 1.0 20080424 EkW initial version
- 1.2 20080902 EkW use PointingProduct

1.258. runPhotometerSmallExtended_pipelinetest

Full Name:	herschel.pacs.spg.pipeline.oldstyle.runPhotometerSmallExtended_pipelinetest
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import runPhotometerSmallExtended_pipelinetest

History

- 1.0 20080424 EkW initial version
- 1.1 20080715 EkW add usage of calTree
- 1.2 20080902 EkW use PointingProduct

1.259. runPhotometerSmallExtended

Full Name:	herschel.pacs.spg.pipeline.oldstyle.runPhotometerSmallExtended
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import runPhotometerSmallExtended

History

- 1.0 20080424 EkW initial version
- 1.1 20080715 EkW add usage of calTree
- 1.2 20080902 EkW use PointingProduct

1.260. runSpectrometerChopNodMap

Full Name:	herschel.pacs.spg.pipeline.oldstyle.runSpectrometerChopNodMap
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import runSpectrometerChopNodMap

History

- 1.0 20080801 JS initial version

1.261. runSpectrometerChopNodStar

Full Name:	herschel.pacs.spg.pipeline.oldstyle.runSpectrometerChopNodStar
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import runSpectrometerChopNodStar

History

- 1.0 20080801 JS initial version

1.262. runSpectrometerFrames

Full Name:	herschel.pacs.spg.pipeline.oldstyle.runSpectrometerFrames
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import runSpectrometerFrames

History

- 1.0 20080430 JS initial version

1.263. runSpectrometerOffMap

Full Name:	herschel.pacs.spg.pipeline.oldstyle.runSpectrometerOffMap
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import runSpectrometerOffMap

History

- 1.0 20080801 JS initial version

1.264. runSpectrometerRamps

Full Name:	herschel.pacs.spg.pipeline.oldstyle.runSpectrometerRamps
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import runSpectrometerRamps

History

- 1.0 20080430 JS initial version

1.265. runSpectrometerWaveSwitch

Full Name:	herschel.pacs.spg.pipeline.oldstyle.runSpectrometerWaveSwitch
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.oldstyle import runSpectrometerWaveSwitch

History

- 1.0 20080801 JS initial version

1.266. scanMapL1

Full Name:	herschel.pacs.spg.pipeline.ipipe.phot.scanMapL1
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.ipipe.phot import scanMapL1

History

- 1.0 20090313 EkW
- 2.0 20100119 EkW Prepare for Operations

1.267. scanMapL2

Full Name:	herschel.pacs.spg.pipeline.ipipe.phot.scanMapL2
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.ipipe.phot import scanMapL2

History

- 1.0 20090313 EkW
- 2.0 20100119 EkW Prepare for operation

1.268. scanMapSimpleL1

Full Name:	herschel.pacs.spg.pipeline.ipipe.phot.scanMapSimpleL1
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.ipipe.phot import scanMapSimpleL1

History

- 1.0 20090313 EkW
- 2.0 20100119 EkW Prepare for Operations

1.269. scanMapSimpleL2

Full Name:	herschel.pacs.spg.pipeline.ipipe.phot.scanMapSimpleL2
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.ipipe.phot import scanMapSimpleL2

History

- 1.0 20090313 EkW
- 2.0 20100120 EkW prepare for operation

1.270. SelectPacsPipeline

Full Name:	herschel.pacs.spg.pipeline.SelectPacsPipeline
Type:	Java Class -
Import:	from herschel.pacs.spg.pipeline import SelectPacsPipeline

History

- 1.0 2009/09/18 JdJ basing on pacsspectro_pipeline.py (JS) / pacsphoto_pipeline.py (EkW)
- 2.0 2009/11/10 EkW adopt to Photometer
- 2.1 2010/02/02 EkW clean up
- 2.2 2010/02/03 EkW Path corrections

1.271. shiftedRamps

Full Name:	herschel.pacs.toolboxes.spec.shiftedRamps
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spec import shiftedRamps

Description

function that shifts ramps onto the reset point

This function loads the according Ramps object, reads the 1D data for all pixels and sets the start point of the ramps by reading the DMC status parameter CRCRMP and the user shift input value. Then the ramp data is returned as a 4D array.

API Summary

Jython Syntax
`Double4d outdata = shiftedRamps(Ramps inRamp [, int userShift])`

Properties
`Ramps inRamp [INPUT, MANDATORY, default=NO default value]`
`int userShift [INPUT, OPTIONAL, default=0]`
`Double4d outdata [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Ramps inRamp [INPUT, MANDATORY, default=NO default value]`
input Ramps object to be shifted

`int userShift [INPUT, OPTIONAL, default=0]`
optional input if the CRCRMP values doesn't show shifts but they are there in real life, default value: 0

`Double4d outdata [OUTPUT, MANDATORY, default=NO default value]`
corrected ramps in Double4d array

See also

- mode CQM test data (dark signals) in which the ramps were shifted

History

- 20050512 JS 1.0 initial version
- 20050524 JS 1.1 imports shifted inside function
- 20050615 JS 1.2 returns a Double4d array instead of Ramps object

- to avoid changing status and mask
- 20050706 JS 1.3 several bug fixes after testing, user shift input possible
- 20050804 JS 1.4 header adopted to jtags syntax
- 20051107 JS 1.5 clean up of imports
- 20060306 JS 1.6 getRampValues() -> getSignal()

1.272. SIGCLIP

Full Name:	herschel.pacs.share.numeric.Sigclip
Alias:	SIGCLIP
Type:	Java Class -
Import:	from herschel.pacs.share.numeric import Sigclip

Description

Sigclip finds values in an array that are more than n*(standard deviation) larger than the next smaller

surrounding value in a box of user defined size. Sigclip replaces these higher values either with the median or the mean of the surrounding values. Sigclip does not include the investigated Pixel for the computation of the median, mean or the sigma The user has the choice of 3 parameters: 1. envSize: the size of the environment of surrounding pixels. If i is the coordinate of the tested pixel, the environment will be [i-envSize, i+envSize] in the 1d case. For multi-dimenional arrays the box is extended in all dimensions. The default value for envSize is 3. 2. nsigma: the number of sigmas that a value has to exceed its next smaller value before it will be replaced. The default value is also 3. 3. The returnmode: Sigclip.RETURN_ARRAY (=0) returns a copy of the input array with the replaced values. Sigclip.RETURN_BOOL (=1) returns a boolean array where the clipped locations are marked as true (the other values are false). 4. the mode: Sigclip.MEAN or Sigclip.MEDIAN determines if median or mean of the neighbouring values is used to replace a sigclipped value

Example

Example 1: apply Sigclip to an Int1d array

```
array = Int1d.range(9)
array.set(5, 20)
print array
[0,1,2,3,4,20,6,7,8]
print array.apply( Sigclip() -)
[0,1,2,3,4,5,6,7,8]
print array.apply( Sigclip(return = Sigclip.RETURN_BOOL) -)
[false,false,false,false,false,true,false,false,false]
```

API Summary

Properties
any 1-3d array of integral type **x** [INPUT, MANDATORY, default=no default value]
the box size that is analysed for every sample **envSize** [INPUT, NOT_MANDATORY, default=3]
the number of sigmas that a value has to exceed the next smaller value to be sigclipped **nsigma** [INPUT, NOT_MANDATORY, default=3]
determines **returnmode** [INPUT, if a numeric or a boolean array is returned, default=NOT_MANDATORY]
determines **mode** [INPUT, if median or mean is used to replace sigclipped values, default=NOT_MANDATORY]

API details

Properties

any 1-3d array of integral type x [INPUT, MANDATORY, default=no default value]

the box size that is analysed for every sample envSize [INPUT, NOT_MANDATORY, default=3]

the number of sigmas that a value has to exceed the next smaller value to be sigclipped nsigma [INPUT, NOT_MANDATORY, default=3]

determines returnmode [INPUT, if a numeric or a boolean array is returned, default=NOT_MANDATORY]

Sigclip.RETURN_ARRAY (=0) returns a copy of the input array with the replaced values.
Sigclip.RETURN_BOOL (=1) returns a boolean array where the clipped locations are marked as true (the other values are false).

determines mode [INPUT, if median or mean is used to replace sigclipped values, default=NOT_MANDATORY]

Sigclip.MEAN (=0) uses the mean of the environment to replace sigclipped values.
Sigclip.MEDIAN (=1) uses the median of the environment to replace sigclipped values.

1.273. sigClip

Full Name:	herschel.pacs.toolboxes.numeric.sigClip
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import sigClip

Description

numeric functions that calculates a sigma clipped array of a double array

and the mean or median of the sigma clipped array sigClip() first computes the standard deviation of the input array values and then discards values that deviate more than nSigma times the standard deviation from the mean or median of the array. sigClipMean() averages the resulting sigma clipped array. sigClipMedian() takes the median of the resulting sigma clipped array.

API Summary

Jython Syntax
`Double1d newArray = sigClip(Double1d array, Double nSigma[, int usemedian = <usemedian>])`
`Double average = sigClipMean(Double1d array, Double nSigma[, int usemedian = <usemedian>])`
`Double median = sigClipMedian(Double1d array, Double nSigma[, int usemedian = <usemedian>])`

Properties
`Double1d array [INPUT, MANDATORY, default=NO default value]`
`Double nSigma [INPUT, MANDATORY, default=NO default value]`
`int usemedian [INPUT, OPTIONAL, default=0]`
`Double1d newArray [OUTPUT, MANDATORY, default=NO default value]`
`Double1d array [INPUT, MANDATORY, default=NO default value]`
`Double nSigma [INPUT, MANDATORY, default=NO default value]`
`int usemedian [INPUT, OPTIONAL, default=0]`
`Double average [OUTPUT, MANDATORY, default=NO default value]`
`Double1d array [INPUT, MANDATORY, default=NO default value]`
`Double nSigma [INPUT, MANDATORY, default=NO default value]`
`int usemedian [INPUT, OPTIONAL, default=0]`
`Double median [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

Double1d array [INPUT, MANDATORY, default=NO default value]
input Double1d array

Double nSigma [INPUT, MANDATORY, default=NO default value]
factor to sigma beyond which the values were discarded

int usemedian [INPUT, OPTIONAL, default=0]

0: mean is used as reference for sigma clipping 1: median is used as reference for sigma clipping

Double1d newArray [OUTPUT, MANDATORY, default=NO default value]

resulting sigma clipped array

Double1d array [INPUT, MANDATORY, default=NO default value]

input Double1d array

Double nSigma [INPUT, MANDATORY, default=NO default value]

factor to sigma beyond which the values were discarded

int usemedian [INPUT, OPTIONAL, default=0]

0: mean is used as reference for sigma clipping 1: median is used as reference for sigma clipping

Double average [OUTPUT, MANDATORY, default=NO default value]

resulting mean value

Double1d array [INPUT, MANDATORY, default=NO default value]

input Double1d array

Double nSigma [INPUT, MANDATORY, default=NO default value]

factor to sigma beyond which the values were discarded

int usemedian [INPUT, OPTIONAL, default=0]

0: mean is used as reference for sigma clipping 1: median is used as reference for sigma clipping

Double median [OUTPUT, MANDATORY, default=NO default value]

resulting median value

History

- 1.0 20050603 PR initial version
- 1.1 20050804 JS header adopted to jtags concept

1.274. skyAngle2ChopPos

Full Name:	herschel.pacs.toolboxes.common.skyAngle2ChopPos
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.common import skyAngle2ChopPos

Description

Convert chopper angle wrt to zero point on sky (arcmin) in chopper digital units

API Summary

Jython Syntax
`Double1d chopperPosition = skyAngle2ChopPos(Double1d skyAngle [,redundant = 0][,CalibrationTree calTree] [,ChopperSkyAngle chopperSkyAngle][,ChopperAngle chopperAngle] [,ChopperAngleRedundant chopperAngleRedundant] [,String model][,FineTime time])`

Properties
`Double1d skyAngle [INPUT, MANDATORY, default=NO default value]`
`int redundant [INPUT, Optional, default=default value: 0]`
`CalibrationTree calTree [INPUT, Optional, default=default value: None]`
`ChopperSkyAngle chopperSkyAngle [INPUT, Optional, default=default value: None]`
`ChopperAngle chopperAngle [INPUT, Optional, default=default value: None]`
`ChopperAngleRedundant chopperAngleRedundant [INPUT, Optional, default=default value: None]`
`String model [INPUT, Optional, default=default value: "FM"]`
`FineTime time [INPUT, Optional, default=default value: present date]`
`Double1d chopperPosition [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

Double1d skyAngle [INPUT, MANDATORY, default=NO default value]
sky angle wrt. to optical zero in arcmin

int redundant [INPUT, Optional, default=default value: 0]
if 0: nominal field plate calibration is used (FPI), if 1: redundant one is used (FPII)

CalibrationTree calTree [INPUT, Optional, default=default value: None]
cal file access

ChopperSkyAngle chopperSkyAngle [INPUT, Optional, default=default value: None]
chopper fpu angle to sky angle cal file access

ChopperAngle chopperAngle [INPUT, Optional, default=default value: None]
chopper calibration readouts to fpu angle cal file access

ChopperAngleRedundant chopperAngleRedundant [INPUT, Optional, default=default value: None]
chopper calibration for redundant FP readouts to fpu angle cal file access

String model [INPUT, Optional, default=default value: "FM"]
model name FM, FS or QM

FineTime time [INPUT, Optional, default=default value: present date]
time of measurement

Double1d chopperPosition [OUTPUT, MANDATORY, default=NO default value]
Chopper position in readout units

See also

- ???

History

- 13/03/2007 JS Creation
- 22/11/2007 JS adaption to new cal framework
- 21/02/2008 JS adapt again to new cal framework
- 25/02/2008 EkW import correction for cal framework
- 12/11/2008 JS remove optional paramter zeroOffset, its in ChopperSkyAngle cal file now

1.275. smallSourceL1

Full Name:	herschel.pacs.spg.pipeline.ipipe.phot.smallSourceL1
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.ipipe.phot import smallSourceL1

History

- 1.0 20090313 EkW
- 2.0 20100120 EkW prepare for operation

1.276. smallSourceL2

Full Name:	herschel.pacs.spg.pipeline.ipipe.phot.smallSourceL2
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.ipipe.phot import smallSourceL2

History

- 1.0 20090313 EkW
- 2.0 20100120 EkW Preparae for Operations

1.277. SpecAddInstantPointingRampsTask

Full Name:	herschel.pacs.toolboxes.spec.SpecAddInstantPointingRampsTask
Type:	Java Task -
Import:	from herschel.pacs.toolboxes.spec import SpecAddInstantPointingRampsTask
Category:	Task/PACS Task

Description

Add PointingProduct to Ramps class

Include s/c pointing as coordinates and additonal pointing information like scanNumber in PACS ramps product. By default the Filtered Pointing information is used, but also the gyro propagated Pointing information may be used. This is done by using the Ramps status entry FINETIME and extract the associated information from the PointingProduct. Also the SIAM matrix is applied and aberration is done (if the proper Products are passed). The result is added to the status entry of the Ramps Product.

API Summary

Jython Syntax
`outRamps = specAddInstantPointingRamps(inRamps , pp [, calTree][, siam][, orbitEphem][, horizons][, isSso][, noInter][, useGyro][, copy])`

Properties
Ramps **inRamps** [INPUT, MANDATORY, default=NO default value]
PointingProduct **pp** [INPUT, MANDATORY, default=NO default value]
PacsCal **calTree** [INPUT, OPTIONAL, default=None]
Siam **siam** [INPUT, OPTIONAL, default=None]
OrbitEphemerisProduct **orbitEpehm** [INPUT, OPTIONAL, default=None]
Horizons **horizons** [INPUT, OPTIONAL, default=None]
Boolean **isSso** [INPUT, OPTIONAL, default=False]
Boolean **noInter** [INPUT, OPTIONAL, default=False]
Boolean **useGyro** [INPUT, OPTIONAL, default=False]
Integer **copy** [INPUT, OPTIONAL, default=None]
Ramps **outRamps** [OUTPUT, MANDATORY, default=NO default value]

API details

Properties

Ramps inRamps [INPUT, MANDATORY, default=NO default value]
Input Ramps class

PointingProduct pp [INPUT, MANDATORY, default=NO default value]
Pointing Product

PacsCal calTree [INPUT, OPTIONAL, default=None]

Pacs Calibration Tree

Siam siam [INPUT, OPTIONAL, default=None]

SIAM Product

OrbitEphemerisProduct orbitEpehm [INPUT, OPTIONAL, default=None]

Orbit Ephemeris Product needed for aberration correction of non SSO objects

Horizons horizons [INPUT, OPTIONAL, default=None]

Horizons Product (Needed for aberration correction of SSO objects)

Boolean isSso [INPUT, OPTIONAL, default=False]

Is it a solar system Object ?

Boolean noInter [INPUT, OPTIONAL, default=False]

Is it a solar system Object ?

Boolean useGyro [INPUT, OPTIONAL, default=False]

Use the Gyro propagated information instead of the filtered

Integer copy [INPUT, OPTIONAL, default=None]

Copy the Ramps class instead of just adding the status word

Ramps outRamps [OUTPUT, MANDATORY, default=NO default value]

Output ramps (default it is the reference of input Ramps + new status entries)

History

- 2009-12-22 - JS: 1.0, Initial version of this Task

1.278. SpecAddInstantPointingTask

Full Name:	herschel.pacs.spg.spec.SpecAddInstantPointingTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.spec import SpecAddInstantPointingTask

Description

Include spacecraft pointing as coordinates in Frames, associates raster point counter,

nod counter or sky line scan counter to every Frame of spectrometer.

By default the Filtered Pointing information is used, but also the gyro propagated pointing information may be used.

This is done by using the Frames status entry FINETIME and extract the associated information from the PointingProduct.

Also the SIAM matrix is applied and aberration is done (if the proper Products are passed).

The result is added to the status entry of the Frames Product.

API Summary

Jython Syntax

`Frames outFrames = specAddInstantPointing(Frames inFrames, PointingProduct pp,`
`[, calTree = calTree][,siam = siam][,orbitEphem=orbitEphem]`
`[,horizons=horizons][,isSso = isSso][,noInter=noInter][,useGyro = useGyro] [,copy = <copy>])`

Properties
`Frames inFrames [INPUT, MANDATORY, default=NO default value]`
`PontingProduct pp [INPUT, MANDATORY, default=NO default value]`
`PacsCal calTree [INPUT, OPTIONAL, default=None]`
`Siam siam [INPUT, OPTIONAL, default=None]`
`OrbitEphemerisProduct orbitEphem [INPUT, OPTIONAL, default=None]`
`Horizons horizons [INPUT, OPTIONAL, default=None]`
`Boolean isSso [INPUT, OPTIONAL, default=False]`
`Boolean noInter [INPUT, OPTIONAL, default=False]`
`Boolean useGyro [INPUT, OPTIONAL, default=False]`
`Integer copy [INPUT, Optional, default=default value: 0]`
`Frames outFrames [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

Frames inFrames [INPUT, MANDATORY, default=NO default value]
Input Frames

PontingProduct pp [INPUT, MANDATORY, default=NO default value]

The Pointing Product covering the same time as the dataframes of the input sequence

PacsCal calTree [INPUT, OPTIONAL, default=None]

Pacs Calibration Tree

Siam siam [INPUT, OPTIONAL, default=None]

SIAM Product

OrbitEphemerisProduct orbitEphem [INPUT, OPTIONAL, default=None]

Orbit Ephemeris Product needed for aberration correction of non SSO objects

Horizons horizons [INPUT, OPTIONAL, default=None]

Horizons Product (Needed for aberration correction of SSO objects)

Boolean isSso [INPUT, OPTIONAL, default=False]

Is it a solar system Object ?

Boolean noInter [INPUT, OPTIONAL, default=False]

interpolate or not?

Boolean useGyro [INPUT, OPTIONAL, default=False]

Use the Gyro propagated information instead of the filtered

Integer copy [INPUT, Optional, default=default value: 0]

indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

Frames outFrames [OUTPUT, MANDATORY, default=NO default value]

Output Frames

History

- 0.1 19-Jul-2006 EkW Initial version of this Task basing on script of BV
- - not for production : for End2End test
- 0.5 10-Jul-2008 EkW First implementation
- 0.6 31-Jul-2008 EkW Add onRaster - offRaster information

1.279. SpecAddNodTask

Full Name:	herschel.pacs.spg.spec.SpecAddNodTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import SpecAddNodTask

Description

This task combines the nod positions for a chop/nod observation

note: for a first implementation: only average Nod A and B of one nod cycle, do not average grating up- and downscan If a noise dataset is available the signal is computed as a weighted mean (WeightedMean class of numerics), and the noise is propagated accordingly If the noise dataset is not available, the signal is computed as a normal mean and a noise dataset is added as the stddev of the mean. The status entries are propagated as a median except of LBL2 and BLOCKIDX columns which are not changed. Masks are propagated with an "or" operation.

API Summary

Jython Syntax

```
Frames outFrame = specAddNod( Frames inFrame [, Integer
useWeightedMean] )
```

Properties

Frames **inFrame** [INPUT, MANDATORY, default=NO default value]

Integer **useWeightedMean** [INPUT, Optional, default=0]

Frames **outFrame** [OUTPUT, MANDATORY, default=NO default value]

API details

Properties

Frames inFrame [INPUT, MANDATORY, default=NO default value]

input Frames object

Integer useWeightedMean [INPUT, Optional, default=0]

if > 0 signal is averaged by a weighted mean otherwise by a standard mean

Frames outFrame [OUTPUT, MANDATORY, default=NO default value]

returned nod combined Frames object

History

- 2008-10-17 - JS: 0.1, initial stub
- 2009-04-21 - JS: 1.0, first version

1.280. SpecAssignRaDecTask

Full Name:	herschel.pacs.spg.spec.SpecAssignRaDecTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import SpecAssignRaDecTask

Description

Assign RA and Dec position for every pixel.

It reads the cal files ArrayInstrument and ModuleArray. If it encounters the PV calibration which distinguishes between the chopper throws small, medium and large, it picks the corresponding spatial calibration for the uniq chopper throw used in the observation. The relative offsets of the spaxel with respect to module 12 is calculated. Then the absolute positions of the spaxels in Ra/Dec on sky is determined by using the jsky.coords.WCSTransform tool of ESO and taking into account the roll angle.

API Summary

Jython Syntax

```
Frames outFrame = specAssignRaDec(Frames inFrame
[,calTree=calTree] [arrayInstrument=arrayInstrument]
[,moduleArray=moduleArray] [,copy=copy] )
```

Properties

- `Frames inFrame [INPUT, MANDATORY, default=NO default value]`
- `PacsCal calTree [INPUT, Optional, default=null]`
- `ArrayInstrument arrayInstrument [INPUT, Optional, default=null]`
- `ModuleArray moduleArray [INPUT, Optional, default=null]`
- `Integer copy [INPUT, OPTIONAL, default=0]`

API details

Properties

`Frames inFrame [INPUT, MANDATORY, default=NO default value]`

input Frames object

`PacsCal calTree [INPUT, Optional, default=null]`

input calibration tree

`ArrayInstrument arrayInstrument [INPUT, Optional, default=null]`

input arrayInstrument cal file

`ModuleArray moduleArray [INPUT, Optional, default=null]`

input moduleArray cal file

`Integer copy [INPUT, OPTIONAL, default=0]`

copy : 0 (default) - give back a reference, 1: give back a copy

History

- 0.1 20080227 JS initial version
- 0.2 20080728 JS updated to use either XY stage data or pointing product
- 1.0 20091103 JS updated using new PV cal files

1.281. SpecAvgPlateauTask

Full Name:	herschel.pacs.spg.spec.SpecAvgPlateauTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import SpecAvgPlateauTask

Description

Average all valid signals on chopper plateaux and treat Status and mask parameter

and discards flagged samples by applying the master mask.

API Summary

Jython Syntax
`outFrame = specAvgPlateau(inFrame[,sigclip=sigclip][,mean=mean][, quality] [,copy=copy] )`

Properties
`Frames inFrame [INPUT, MANDATORY, default=No default value]`
`Integer sigclip [INPUT, OPTIONAL, default=default value:0]`
`Integer mean [INPUT, OPTIONAL, default=default value: 0]`
`QualityContext quality [INPUT, OPTIONAL, default=default value: null]`
`Integer copy [INPUT, OPTIONAL, default=default value: 0]`
`Frames outFrame [OUTPUT, MANDATORY, default=No default value]`

API details

Properties

`Frames inFrame [INPUT, MANDATORY, default=No default value]`

input Frames object

`Integer sigclip [INPUT, OPTIONAL, default=default value:0]`

Sigma Clipping value 0: no sigma clipping WARNING: this option should only be used for long chopper plateaux (more than 3 readouts per plateau)

`Integer mean [INPUT, OPTIONAL, default=default value: 0]`

0 : use MEAN, 1 : MEDIAN

`QualityContext quality [INPUT, OPTIONAL, default=default value: null]`

quality context

`Integer copy [INPUT, OPTIONAL, default=default value: 0]`

Copy the input (1) and generate new output or overwrite the input (0)

Frames outFrame [OUTPUT, MANDATORY, default=No default value]
returned Frames object rebinned on chopper plateaux

History

- `2008-06-16 - JS`: 1.0 initial prototype version based on PhotAvgPlateauTask
- `2008-06-19 - JS`: 1.1 resampling of wavelength dataset introduced
- `2008-06-20 - JS`: 1.2 resampling of StDev and Ra and Dec datasets introduced
- `2008-07-24 - JS`: 1.3 flexible mask handling introduced, complete refactoring
- `2008-10-10 - JS`: 1.4 implementation of SPR 1080

1.282. SpecConvDigit2VoltsPerSecFramesTask

Full Name:	herschel.pacs.spg.spec.SpecConvDigit2VoltsPerSecFramesTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.spec import SpecConvDigit2VoltsPerSecFramesTask

Description

This task converts the values coming from SPU-on-board fitting or the pipeline task fitRamps to V/s.

Information for Project Scientists: This module task is a combination of former used DigitsPerRampLen2DigitsPerSecTask.py and SpecConvDigit2VoltsFramesTask.py.

The conversion for SPU signal values in Frames depends on the reset length. The ramp fitting takes the length of the ramp it fits as unity. The slope numbers it produces are therefore digital units per ramp length. First this module converts Digits/reset interval to Digits/sec for the Spectrometer Frames.

framesSignal = frameSignal * 256Hz/RampLength

RampLength is the number of readouts per Ramp.

Then this task converts the digits/sec to Volts/sec using the corresponding calibration file "SpecVolts". This implies that the resulting values are converted to positive signals.

framesSignal = -frameSignal * (endVolt - startVolt) / (endDigit - startDigit)

API Summary

Jython Syntax
`outFrames = specConvDigit2VoltsPerSecFrames(Frames inFrames [, calTree][,readouts2Volts][, int copy = <copy>])`

Properties
`Frames inFrames [INPUT, MANDATORY, default=NO default value]`
`Frames outFrames [OUTPUT, MANDATORY, default=NO default value]`
`CalibrationTree calTree [INPUT, Optional, default=default value: None]`
`Readouts2Volts readouts2Volts [INPUT, Optional, default=default value: None]`
`Integer copy [INPUT, Optional, default=default value: 0]`

API details

Properties

Frames inFrames [INPUT, MANDATORY, default=NO default value]
Detector data are converted from digits/reset interval to digits/volts

Frames outFrames [OUTPUT, MANDATORY, default=NO default value]
Depending on copy keyword a copy of the Frames class with converted units or an updated Frames class

CalibrationTree calTree [INPUT, Optional, default=default value: None]
cal file access

Readouts2Volts readouts2Volts [INPUT, Optional, default=default value: None]
cal file access

Integer copy [INPUT, Optional, default=default value: 0]
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

History

- 0.1 23-March-2006 RV Initial version of this Task
- 0.2 14-July-2006 RV Quantity conversion included
- 0.3 19-Jul-2006 EkW Comment out Quantity issues : Usage of Quantities is still not agreed/defined
- 1.0 21-Aug-2007 JS DigitsPerRampLen2DigitsPerSec and SpecConvDigit2VoltsFrames combined to one
- task SpecConvDigit2VoltsPerSecFrames
- 1.1 23-Aug-2007 JS made the signals positive
- 1.2 21-Nov-2007 JS conversion to new cal framework
- 1.3 21-Feb-2008 JS adapt again to new cal framework

1.283. SpecConvDigit2VoltsRampsTask

Full Name:	herschel.pacs.spg.spec.SpecConvDigit2VoltsRampsTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.spec import SpecConvDigit2VoltsRampsTask

Description

Convert Digits to Volts for the Spectrometer Ramps

and multiply by -1 to have positive signals after the pipeline task fitRamps

API Summary

Jython Syntax
`outRamps = specConvDigit2VoltsRamps(Ramps inRamps[, calTree] [,Readouts2Volts][, int copy = <copy>])`

Properties
`Ramps inRamps [INPUT, MANDATORY, default=NO default value]`
`Ramps outRamps [OUTPUT, MANDATORY, default=NO default value]`
`CalibrationTree calTree [INPUT, Optional, default=default value: None]`
`Readouts2Volts readouts2Volts [INPUT, Optional, default=default value: None]`
`Integer copy [INPUT, Optional, default=default value: 0]`

API details

Properties

`Ramps inRamps [INPUT, MANDATORY, default=NO default value]`

Detector data are converted from digits to volts and negatived to have positive signals after FitRamps

`Ramps outRamps [OUTPUT, MANDATORY, default=NO default value]`

Depending on copy keyword a copy of the Ramps class with converted digits to Volts or an updated Ramps class

`CalibrationTree calTree [INPUT, Optional, default=default value: None]`

cal file access

`Readouts2Volts readouts2Volts [INPUT, Optional, default=default value: None]`

cal file access .

`Integer copy [INPUT, Optional, default=default value: 0]`

indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

History

- 1.0 08-Mar-2006 EkW Initial version of this Task
- 1.1 08-Nov-2006 Ekw Readouts -> Signal
- 1.2 05-Dec-2006 JS implementation of SPR 635 (use Ramps.getSignal/setSignal)
- 1.3 06-Dec-2006 JS quantity added for TRamps
- 1.4 21-Aug-2007 JS convert nanoTitan nT lib to Unit Api
- 1.5 21-Nov-2007 JS convert to new cal framework
- 1.6 07-Dec-2007 JS performance improvement
- 1.7 21-Feb-2008 JS adapt again to new cal framework

1.284. SpecCorrectCrosstalkTask

Full Name:	herschel.pacs.spg.spec.SpecCorrectCrosstalkTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import SpecCorrectCrosstalkTask

Description

JAVA Task as it need loops

This task reads crosstalk ratios from a calibration file and subtracts from every pixel the signal fraction of a crosstalking pixel. The structure of the calibration file ## The calibration file should be created and edited with the script makeCalCrosstalkMatrix.py. The Calfile contains an ArrayDataset with two Double2d arrays in it, one for the red and one for the blue channel. Dimensions of the Double2d are [number of cross-talks found, 5). The values are composed like this: [k,0] : the row value of the original pixel [k,1] : the column value of the original pixel [k,2] : the row value of the crosstalking pixel [k,3] : the column value of the crosstalking pixel [k,4] : the crosstalking ratio

Example

Example 1: ##########

```
lets assume the signal of pixel (4,7) appears with 50% in pixel (0,0) and
the signal of pixel (8, 10) appears with 13% in pixel (5,1). Then the array
looks like this:
[0,0] = 4
[0,1] = 7
[0,2] = 0
[0,3] = 0
[0,4] = 0.5
[1,0] = 8
[1,1] = 10
[1,2] = 5
[1,3] = 1
[1,4] = 0.13
Thus the crosstalk corrected values of pixel (0,0) and (5,1) would be
realsig(0,0) = realsig(0,0) -- 0.5*sig(4,7)
realsig(5,1) = realsig(5,1) -- 0.13*sig(8,10)
These simple subtractions are applied by this task for the whole signal array.
```

API Summary

Jython Syntax

```
Frames outFrame = specCorrectCrosstalk(Frames inFrame [, PacsCal
calTree]
[,CrosstalkMatrix crosstalkMatrix][, int copy][, QualityContext
qualityContext])
```

Properties

Frames **inFrame** [INPUT, MANDATORY, default=NO default value]
PacsCal **calTree** [INPUT, Optional, default=default value: None]
CrosstalkMatrix **crosstalkMatrix** [INPUT, Optional, default=default value: None]
Integer **copy** [INPUT, Optional, default=default value: 0]

Properties
QualityContext **qualityContext** [INOUT, Optional, default=default value: None]
Frames **outFrame** [OUTPUT, MANDATORY, default=NO default value]

API details

Properties

Frames inFrame [INPUT, MANDATORY, default=NO default value]
input Frames object

PacsCal calTree [INPUT, Optional, default=default value: None]
cal file access

CrosstalkMatrix crosstalkMatrix [INPUT, Optional, default=default value: None]
crosstalk matrix cal file

Integer copy [INPUT, Optional, default=default value: 0]
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

QualityContext qualityContext [INOUT, Optional, default=default value: None]
quality control flag

Frames outFrame [OUTPUT, MANDATORY, default=NO default value]
returned Frames object containing the cross talk corrected signals

History

- 1.0 20080409 JS adopted from PhotCorrectCrosstalkTask

1.285. SpecCorrectHerschelVelocityTask

Full Name:	herschel.pacs.spg.spec.SpecCorrectHerschelVelocityTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import SpecCorrectHerschelVelocityTask

Description

Corrects the wavelength values for the line-of-sight velocity.

Additionally the spacecraft velocity in the direction of the pointing is stored in the Status column "VSC".

This task uses the HCSS RadialVelocityTask to compute the radial velocity. The corrections Sun's v wrt LSR and Earth's v wrt Sun are applied. All wavelengths are corrected as follows: wlen = wlen_observed*(1-vsc/c)

API Summary

Properties
Frames **inFrames** [INPUT, MANDATORY, default=no default value]
OrbitEphemerisProduct **oep** [INPUT, MANDATORY, default=no default value]
PointingProduct **pp** [INPUT, MANDATORY, default=no default value]

API details

Properties

Frames inFrames [INPUT, MANDATORY, default=no default value]

The input Frames product

OrbitEphemerisProduct oep [INPUT, MANDATORY, default=no default value]

The orbit ephemeris product corresponding to the observation

PointingProduct pp [INPUT, MANDATORY, default=no default value]

The pointing product corresponding to the observation

1.286. SpecCorrectSignalNonLinearitiesTask

Full Name:	herschel.pacs.spg.spec.SpecCorrectSignalNonLinearitiesTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import SpecCorrectSignalNonLinearitiesTask

Description

This task reads the NonLinearity calibration file, and corrects (linearizes) the signals of each pixel

by calculating a second order polynomial using the coefficients from the cal file.

The structure of the calibration file:

the cal file is organized in 2 Double3d(18, 25, 3) arrays for the red and blue array containing the 3 polynomial coefficients.

API Summary

Jython Syntax
`Frames outFrame = specCorrectSignalNonLinearities(Frames inFrame [, PacsCal calTree]` `[,NonLinearity nonLinearity][, int copy])`

Properties
`Frames` **`inFrame`** `[INPUT, MANDATORY, default=NO default value]`
`PacsCal` **`calTree`** `[INPUT, Optional, default=default value: None]`
`NonLinearity` **`nonLinearity`** `[INPUT, Optional, default=default value: None]`
`Integer` **`copy`** `[INPUT, Optional, default=default value: 0]`
`Frames` **`outFrame`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
input Frames object

`PacsCal calTree [INPUT, Optional, default=default value: None]`
cal file access

`NonLinearity nonLinearity [INPUT, Optional, default=default value: None]`
polynomial coefficients cal file

`Integer copy [INPUT, Optional, default=default value: 0]`
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

Frames outFrame [OUTPUT, MANDATORY, default=NO default value]
returned Frames object containing linearised signals

History

- `2008-07-17 - JS`: 1.0: initial version

1.287. SpecDiffChopTask

Full Name:	herschel.pacs.spg.spec.SpecDiffChopTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import SpecDiffChopTask

Description

Subract Off Source Signals from On Source Signals. A pairwise differencing algorithm is used on all individual

frames of the plateaux. This task first scans the data and divides it blocks with contstant grating. Then it looks at how the chopper position changes from A -> B to determine which pattern is used (ABAB or ABBA) and with how many repetitions within a grating position. In case of ABBA the B plateaux are split. After determining all the plateaux the indices of the frames which need to be subtracted from each other are determined. A and B are assigned to ON or OFF depending on the nodding position. Then the output frame is created and the data is filled as follows:

- The signal dataset with ON-OFF
- The masks as ON|OFF, also if one of the datapoints is masked then the result will be masked.
- The other datasets are filled with the ON values
- The Status of the ON position.
- The labels of both ON and OFF are merged in a new column LBL2
- The reset indices of both ON and OFF source are stored in the columns RESETINDEX and OFF_RESETIDX

This task needs the GRATMOV mask to determine the boundaries between the grating positions. Data with GRATMOV flagged does not appear in the output frames. Otherwise, all other masks are just propagated. This task does not check for active masks.

API Summary

Properties
Frames **framesIn** [INPUT, MANDATORY, default=no default value]
Boolean **removeCalSrc** [INPUT, OPTIONAL, default=true]
Boolean **normalize** [INPUT, OPTIONAL, default=false]
Frames **framesOut** [OUTPUT, MANDATORY, default=no default value]

API details

Properties

Frames framesIn [INPUT, MANDATORY, default=no default value]

The input Frames product

Boolean removeCalSrc [INPUT, OPTIONAL, default=true]

Remove the calibration blocks if true

Boolean normalize [INPUT, OPTIONAL, default=false]
Compute 2(A-B)/(A+B) to normalize the fluxes if true. EXPERIMENTAL EXPERT OPTION!

Frames framesOut [OUTPUT, MANDATORY, default=no default value]
The result Frames product with background subtracted signal values

1.288. SpecDiffCsTask

Full Name:	herschel.pacs.spg.spec.SpecDiffCsTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.spec import SpecDiffCsTask

Description

Task computing the absolute response and the dark current of the current observation.

The absolute response and the dark current are computed with the data coming from the calibration block of the current observation and from calibration tables.

During the execution of the calibration block, the chopper quickly moves back and forth from one internal calibration source (CS1) to the other (CS2). These two internal calibration sources have different temperatures.

In this task, we compute the average of the signal pairwise differences between the two calibration sources. By dividing this average by the known relative spectral response multiplied by the difference of the spectral flux densities of the calibration sources, we get the absolute pixel response.

As an additive term in the signal expression, the dark current is then deduced from the difference between the observed signal and the theoritical signal computed from the spectral flux densities of the calibration sources, the relative spectral response and the absolute pixel response.

As there are three different bands in the blue channel, the responses in these three bands are computed even if only one band was selected for the execution of the calibration block.

The errors on the signal values are propagated all along the computation and the errors on the response(s) and on the dark are also returned in the output product.

Example

Example 1: from Hipe:

```
from herschel.pacs.spg.spec import SpecDiffCsTask
# We activate some masks: -"UNCLEANCHOP" to discard data during a chopper
movement, -"GLITCH" to discard data affected by a glitch.
frame = activateMasks(frame, String1d(["UNCLEANCHOP", -"GLITCH"]), exclusive
= True)
# We compute the response and the dark from the measurements of the
calibration block of the current observation.
response_and_dark = specDiffCs(frame)
```

API Summary

Jython Syntax

```
SpecCsResponseAndDarkProduct CsResponseAndDark = specDiffCs(Frames
inFrames [, PacsCal calTree=calTree] [, CalSourceFluxProduct
calSourceFluxProduct=None] [, RelCalSourceFluxProduct
relCalSourceFluxProduct=None])
```

Properties

```
Frames inFrames [INPUT, MANDATORY, default=NO default value]
```

Properties
Optional **calTree** [PacsCal, default value: calTree, default=no default value]
Optional **calSourceFluxProduct** [CalSourceFluxProduct, default value: None, default=no default value]
Optional **relCalSourceFluxProduct** [RelCalSourceFluxProduct, default value: None, default=no default value]

API details

Properties

Frames inFrames [INPUT, MANDATORY, default=NO default value]
Input Frames

Optional calTree [PacsCal, default value: calTree, default=no default value]
Calibration tree

Optional calSourceFluxProduct [CalSourceFluxProduct, default value: None, default=no default value]
Calibration Source Flux Product which contains the absolute fluxes of the internal calibration sources at the primary key wavelengths.

Optional relCalSourceFluxProduct [RelCalSourceFluxProduct, default value: None, default=no default value]
Relative Calibration Source Flux Product which contains the relative flux ratios of the internal calibration sources between secondary and primary key wavelengths.

See also

- ???

History

- 1.0 15/10/2008 CJ Initial version
- 1.6 21/04/2009 CJ Response and dark computed, use of the Calibration Products, units added, bugs fixed
- 1.10 18/08/2009 CJ New version. Added OBCP 35 support.
- 1.16 18/11/2009 CJ Update of the User's Reference Manual documentation

1.289. SpecEstimateNoiseTask

Full Name:	herschel.pacs.spg.spec.SpecEstimateNoiseTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.spec import SpecEstimateNoiseTask

Description

Jython task that estimates the noise at level1 for each pixel and fills the Noise dataset.

First it selects the frames according to chopperplateau position 1, 2 or 0. Then it subtracts the median filtered signal (using bins with binWidth width) and calculates the standard deviation in bins with binWidth width by discarding the readouts masked by the Master mask since they could propagate and fake very high noise in the neighboring readouts. The noise in the masked readouts is set to the square root of the according signal afterwards.

API Summary

Jython Syntax
`Frames frames = specEstimateNoise(Frames frames [,Integer binWidth][,copy] )`

Properties
`Frames frames [INOUT, MANDATORY, default=NO default value]`
`Integer binWidth [INPUT, OPTIONAL, default=default value :5]`
`String copy [INPUT, OPTIONAL, default=default value : 0]`

API details

Properties

`Frames frames [INOUT, MANDATORY, default=NO default value]`
input/output Frames object

`Integer binWidth [INPUT, OPTIONAL, default=default value :5]`
width of the median filter and stddev calculation bin

`String copy [INPUT, OPTIONAL, default=default value : 0]`
Copy the input (0) and generate new output or overwrite the input (1)

History

- 1.0 20091022 JS initial version

1.290. SpecExtendStatusTask

Full Name:	herschel.pacs.spg.spec.SpecExtendStatusTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import SpecExtendStatusTask

Description

This task extends the Status of Frames with the parameters GRATSCAN, CHOPPER, CHOPPOS (see SPR 842).

GRATSCAN counts the number of up- (positive) and down (negative) grating scans.

CHOPPER adds CHOPPERPLATEAU and CALSOURCE.

CHOPPOS indicates the chopper throw (CS, small, medium, large, center, or NoName).

API Summary

Jython Syntax
`Frames outFrame = specExtendStatus(Frames inFrame [, PacsCal calTree]` `[,ChopperThrowDescription][, int copy])`

Properties
`Frames` **`inFrame`** `[INPUT, MANDATORY, default=NO default value]`
`PacsCal` **`calTree`** `[INPUT, Optional, default=None]`
`ChopperThrowDescription` **`chopperThrowDescription`** `[INPUT, Optional, default=None]`
`Integer` **`copy`** `[INPUT, Optional, default=0]`
`Frames` **`outFrame`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Frames inFrame [INPUT, MANDATORY, default=NO default value]`

input Frames object

`PacsCal calTree [INPUT, Optional, default=None]`

cal file access

`ChopperThrowDescription chopperThrowDescription [INPUT, Optional, default=None]`

verbal description of different chopper deflections cal file

`Integer copy [INPUT, Optional, default=0]`

indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

Frames outFrame [OUTPUT, MANDATORY, default=NO default value]
returned Frames object containing the additional status parameters

History

- `2008-07-15 - JS`: 1.0: initial version

1.291. SpecFitSignalDriftTask

Full Name:	herschel.pacs.spg.spec.SpecFitSignalDriftTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.spec import SpecFitSignalDriftTask
Category:	PACS/Spectrometer/Task

Description

Computes how the pixel response changes with time

This task computes how the reponse changes with time by analyzing the background signal from a given Frames object. The initial response R0 is derived with the specDiffCs task from the calibration sources. This task fits a background spectrum to all the off-source points together, after deselecting actively masked datapoints. Therefore, one must (de)activate the proper masks before running this task.

API Summary

Jython Syntax
`framesOut = specFitSignalDrift(framesIn,respDark)`

Properties
`Frames framesIn [INPUT, MANDATORY, default=no default value]`
`SpecCsResponseAndDark respDark [INPUT, MANDATORY, default=no default value]`
`Integer calBlock [INPUT, OPTIONAL, default=0]`
`SpecResponseDrift respDrift [OUTPUT, MANDATORY, default=no default value]`

API details

Properties

`Frames framesIn [INPUT, MANDATORY, default=no default value]`

Science data after execution of at least the tasks waveCalc, rsrfCal and specSubtractDark

`SpecCsResponseAndDark respDark [INPUT, MANDATORY, default=no default value]`

result of averaging the columns of the table.

`Integer calBlock [INPUT, OPTIONAL, default=0]`

The calibration block for which the initial response is obtained (The respDark product may be the result of several calibration blocks)

`SpecResponseDrift respDrift [OUTPUT, MANDATORY, default=no default value]`

Response as function of pixel and time

1.292. SpecFlagBadPixelsFramesTask

Full Name:	herschel.pacs.spg.spec.SpecFlagBadPixelsFramesTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.spec import SpecFlagBadPixelsFramesTask

Description

Jython task that sets the flags for bad pixels ("BADPIXELS" mask)

and noisy pixels ("NOISYPIXELS" mask). The bad/noise pixel coordinates come from the cal files BadPixelMask and NoisyPixelMask

API Summary

Jython Syntax
`Frames frames = specFlagBadPixelsFrames(Frames frames[, CalibrationTree calTree][, BadPixelMask badPixelMask][, NoisyPixelMask noisyPixelMask] [,copy=0] )`

Properties
`Frames frames [INOUT, MANDATORY, default=NO default value]`
`CalibrationTree calTree [INPUT, OPTIONAL, default=default value :None]`
`BadPixelMask badPixelMask [INPUT, OPTIONAL, default=default value :None]`
`NoisyPixelMask noisyPixelMask [INPUT, OPTIONAL, default=default value :None]`
`String copy [INPUT, OPTIONAL, default=default value : 0]`

API details

Properties

Frames frames [INOUT, MANDATORY, default=NO default value]
input/output Frames object

CalibrationTree calTree [INPUT, OPTIONAL, default=default value :None]
Calibration Tree

BadPixelMask badPixelMask [INPUT, OPTIONAL, default=default value :None]
direct input of cal file

NoisyPixelMask noisyPixelMask [INPUT, OPTIONAL, default=default value :None]
direct input of cal file

String copy [INPUT, OPTIONAL, default=default value : 0]
Copy the input (0) and generate new output or overwrite the input (1)

History

- 1.0 20070807 JS initial version
- 1.1 20071121 JS conversion to new cal framework
- 1.2 20080221 JS adapt again to new cal framework

1.293. SpecFlagBadPixelsRampsTask

Full Name:	herschel.pacs.spg.spec.SpecFlagBadPixelsRampsTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.spec import SpecFlagBadPixelsRampsTask

Description

Jython task that sets the flags for bad pixels ("BADPIXELS" mask).

The bad pixel coordinates come from the BadPixelMask cal file.

API Summary

Jython Syntax
`Ramps ramps = specFlagBadPixelsRamps(Ramps ramps[,CalibrationTree calTree][,BadPixelMask badPixelMask] [,copy=0] )`

Properties
`Ramps ramps [INOUT, MANDATORY, default=NO default value]`
`CalibrationTree calTree [INPUT, OPTIONAL, default=default value :None]`
`BadPixelMask badPixelMask [INPUT, OPTIONAL, default=default value :None]`
`String copy [INPUT, OPTIONAL, default=default value : 0]`

API details

Properties

`Ramps ramps [INOUT, MANDATORY, default=NO default value]`
input/output Ramps object

`CalibrationTree calTree [INPUT, OPTIONAL, default=default value :None]`
Calibration Tree

`BadPixelMask badPixelMask [INPUT, OPTIONAL, default=default value :None]`
direct input of cal file###

`String copy [INPUT, OPTIONAL, default=default value : 0]`
Copy the input (0) and generate new output or overwrite the input (1)

History

- 1.0 20070807 JS initial version
- 1.1 20071121 JS conversion to new cal framework

- 1.2 20080221 JS adapt again to new cal framework

1.294. SpecFlagGlitchFramesQTestTask

Full Name:	herschel.pacs.spg.spec.SpecFlagGlitchFramesQTestTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import SpecFlagGlitchFramesQTestTask

Description

Task that masks the Responsivity Jumps (GLITCH) due to glitches at the slope level (i.e. in Frames)

This function works on the time sequence of the slopes (sl), for each pixel individually. The basic algorithm is

- compute 2 differential signals : dsl = sl[i] - sl[i-1] & ds2 = sl[i+1] - sl[i-1]
- compute 2 contrast functions based on q-tests of dsl & ds2 (q1 & q2 respectively) For each of these, the principle is to rune the q-test over a 'box' containing a fixed number of data points (width w) and to slide the box over the whole time sequence. Each data point, i.e. each slope, gets w values of the q-score, one for each box in which it is contained The contrast is taken as the highest of those values.
- apply some thresholding, specific to the kind of events produced by glitches & responsivity jumps (spikes w/ & w/o decays, and staircases) There are different values of the threshold currently used : t0, t1, t2, t3

  For any detector, a given slope i is marqued as affected by a Responsivity Jump in the following cases

  1. q1[i] > t0 & i-1 not flagged as jump

     This identifies the strongest events (computationally) fast and in cases where they would be missed because of other strong events close-by.

  2. q1[i] or q1[i+1] > t1

     q2[i-1] or q2[i+1] > t2

     q2[i-1] and q2[i+1] > t3

     q2[i-1] and q2[i+1] have opposite sign

     This identifies the 'isolated' events, i.e. the classical spikes affecting only one ramp, with weak or no long term effect on the responsivity (more typical for Low Stress detectors)

  3. q1[i] > t1

     q2[i] or q2[i-1] > t2

     q2[i] and q2[i-1] > t3

     q2[i-1] and q2[i+1] have identical sign

     This identifies the 'step' events, i.e. those leading to a long term modification of the responsivity (more typical for High Stress detectors)

  4. both neighbours flagged as event

     This picks up the pathologic events 'hiding' between 2 spikes, sometimes missed because the method is based on ds2

API Summary

Jython Syntax
`Frames outFrames = specFlagGlitchFramesQTestTask(Frames inFrames, Integer qtestwidth, Double1d thresholds)`

Properties
`Frames` **`inFrames`** `[INPUT, MANDATORY, default=NO default value]`
`String` **`copy`** `[INPUT, OPTIONAL, default=default value : 0]`
`Integer` **`qtestwidth`** `[INPUT, OPTIONAL, default=16]`
`Double1d` **`thresholds`** `[INPUT, OPTIONAL, default='Double1d([2.]`
`Integer` **`qtestlow`** `[INPUT, OPTIONAL, default=3]`
`Integer` **`qtesthigh`** `[INPUT, OPTIONAL, default=3]`
`Frames` **`outFrames`** `[OUTPUT, MANDATORY, default=NO default value]`

Miscellaneous

Examples of use of this GLITCH-flagging algorithm can be found in PICC-KL-TN-023

API details

Properties

`Frames inFrames [INPUT, MANDATORY, default=NO default value]`
Frames object

`String copy [INPUT, OPTIONAL, default=default value : 0]`
Copy the input (1) and generate new output or overwrite the input (0)

`Integer qtestwidth [INPUT, OPTIONAL, default=16]`
Width of the sliding box over which q-test is applied

`Double1d thresholds [INPUT, OPTIONAL, default='Double1d([2.]`
The four thresholds to be applied on the q-contrast functions

`Integer qtestlow [INPUT, OPTIONAL, default=3]`
Number of lowest data points excluded from the range computation the q-tests

`Integer qtesthigh [INPUT, OPTIONAL, default=3]`
Number of highest data points excluded from the range computation the q-tests

`Frames outFrames [OUTPUT, MANDATORY, default=NO default value]`
Returned averaged Frames object

History

- 1.0 28-Aug-2007 PR Creation
- 1.1 ..

1.295. SpecFlagOutliersTask

Full Name:	herschel.pacs.spg.spec.SpecFlagOutliersTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import SpecFlagOutliersTask

Description

Help ...

Examples

Example 1: For diagnostics we want to make a rebinned cube using just the OUTLIERS mask only.

```
waveGrid = wavelengthGrid (cube)
activeMasks = String1d ()
activeMasks.append("BLINDPIXELS")
activeMasks.append("BADPIXELS")
activeMasks.append("BADFITPIX")
activeMasks.append("UNCLEANCHOP")
activeMasks.append("GRATMOVE")
activeMasks.append("SATURATION")
activeMasks.append("GLITCH")
cube = activateMasks(cube,masks=activeMasks,exclusive=1)
# Print out active masks for clarity.
print cube.mask.activeMaskTypes
cube = specFlagOutliers(cube,waveGrid,nSigma=3,nIter=2)
# Create a list containing only the one mask.
oneMask = String1d("OUTLIERS")
cube = activateMasks(cube,masks=oneMask),exclusive=1)
rebinnedCube = specWaveRebin(cube,waveGrid)
```

Example 2: Suppose you want to see what the effect of one single mask is one you

```
PacsCube data before rebining.
oneMask = String1d (["OUTLIERS"])
cube = activateMasks(cube,masks=oneMask,exclusive=1)
m=2
n=2 # Central pixel
index=cube.getUnmaskedIndices(n,m)
p=PlotXY(cube.wave[index,n,m],cube.flux[index,n,m])
# Note: If you are doing this while running an interactive pipeline,
# don't forget to re-activate masks that you need for future pipeline
# tasks.
```

Example 3: Flag outliers, saving statistics in PacsCube. Plot standard deviation over wavelength.

```
cube =
specFlagOutliers(cube,waveGrid,nSigma=5.0,ignoreMasks=ignoreMasks,nIter=3,saveStats=1)
wg = waveGrid.wavelengthGrid
stdev = cube["fluxStdev"].data
stdev22 = sv[:,2,2]
p = PlotXY(wg,stdev22)
p.style.line=0
```

Example 4: Get the mean, standard deviation, median and the number of contributors

```
for each wavelength bin as Double1d:
   mean = cube["fluxMean"]
```

Example 4: Get the mean, standard deviation, median and the number of contributors

```
stdev = cube["fluxStdev"]
median = cube["fluxMedian"]
contributors = cube["contributors"]
```

API Summary

Properties
PacsCube **pacsCube** [INPUT, MANDATORY, default=NO default value]
WaveLengthGrid **wavelengthGrid** [INPUT, MANDATORY, default=NO default value]
Double **nSigma** [INPUT, OPTIONAL, default=default value = 5.0]
Integer **nIter** [INPUT, OPTIONAL, default=default value = 1]
Boolean **saveStats** [INPUT, OPTIONAL, default=default value = false.]
PacsCube **PacsCube** [OUTPUT, MANDATORY, default=NO default value]

API details

Properties

PacsCube pacsCube [INPUT, MANDATORY, default=NO default value]

input PacsCube object

WaveLengthGrid wavelengthGrid [INPUT, MANDATORY, default=NO default value]

input wavelengthGrid

Double nSigma [INPUT, OPTIONAL, default=default value = 5.0]

clip at this n * sigma.

Integer nIter [INPUT, OPTIONAL, default=default value = 1]

Number of additional iteractions. Note: nIter = 1 means one pass and one additional iteration.

Boolean saveStats [INPUT, OPTIONAL, default=default value = false.]

Save the fluxStdev, fluxMedian

PacsCube PacsCube [OUTPUT, MANDATORY, default=NO default value]

returned cube with new "OUTLIERS" mask (NW x 5 x 5).

1.296. SpecFlagSaturationFramesTask

Full Name:	herschel.pacs.spg.spec.SpecFlagSaturationFramesTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.spec import SpecFlagSaturationFramesTask

Description

Jython task that sets the flags for saturated signals (populates SATURATION mask),

say very high signal values that could run into saturation.

The saturation of signals depends on the used reset interval, which should be scaled to the reference of 1.0 s used in the cal file. It is assumed that the saturation signal scales linear with reset interval, implying a linear ramp, which is only an approximation, see e.g. debiasing effects.

This module bases on digits/sec values for signals.

Additionally, there is the possibility to check the saturation using the raw ramp pixel, if you input the raw ramp at the corresponding optional parameter rawRamp. This fills the boolean Status column "RAWSAT" (true if saturation detected) and the additional deactivated Mask "RAWSATURATION" which is true for all pixels of a reset interval if the raw ramp saturates!

API Summary

Jython Syntax
`Frames frames = specFlagSaturationFrames(Frames frames [, Ramps rawRamp][, CalibrationTree calTree][, SignalSatLimits signalSatLimits] [,copy] )`

Properties
`Frames frames [INOUT, MANDATORY, default=NO default value]`
`Ramps rawRamp [INPUT, OPTIONAL, default=default value :None]`
`CalibrationTree calTree [INPUT, OPTIONAL, default=default value :None]`
`SignalSatLimits signalSatLimits [INPUT, OPTIONAL, default=default value :None]`
`String copy [INPUT, OPTIONAL, default=default value : 0]`

API details

Properties

Frames frames [INOUT, MANDATORY, default=NO default value]
input/output Frames object

Ramps rawRamp [INPUT, OPTIONAL, default=default value :None]
raw Ramp object

CalibrationTree calTree [INPUT, OPTIONAL, default=default value :None]
Calibration

SignalSatLimits signalSatLimits [INPUT, OPTIONAL, default=default value :None]
Cal file

String copy [INPUT, OPTIONAL, default=default value : 0]
Copy the input (0) and generate new output or overwrite the input (1)

History

- 2007-08-28 - JS: 1.0: initial version
- 2007-11-20 - JS: 1.1: adaption to new cal framework
- 2008-02-21 - JS: 1.2: again adaption to new cal framework
- 2008-03-11 - JS: 1.3: adapt to changes of cal file SignalSatLimits
- 2009-08-06 - JS: 1.4: SCR 1920: add raw ramp saturation flagging

1.297. SpecFlagSaturationRampsTask

Full Name:	herschel.pacs.spg.spec.SpecFlagSaturationRampsTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import SpecFlagSaturationRampsTask

Description

task that sets the flags for saturated ramp readouts (populates SATURATION mask).

It extrapolates averaged ramps and checks for saturation.

Additionally, it checks for saturation of the raw Ramp of the high responsivity pixel, and sets a Status flag "RAWSAT" to true if saturated, if you input the raw ramp at the corresponding optional parameter rawRamp. Then it also adds a deactivated mask called "RAWSATURATION" which is true for all pixels of a reset where the raw ramp is saturated.

API Summary

Jython Syntax
`Ramps outramp = specFlagSaturationRamps(Ramps inRamp [, rawRamp = rawRamp]`
`[,calTree=calTree][,rampSatLimits=rampSatLimits] [, int copy = 0] [, qualityContext =`
`qualityContext] )`

Properties
`Ramps` **`inRamp`** `[INPUT, MANDATORY, default=NO default value]`
`Ramps` **`rawRamp`** `[INPUT, Optional, default=default value: null]`
`PacsCal` **`calTree`** `[INPUT, Optional, default=default value: null]`
`RampSatLimits` **`rampSatLimits`** `[INPUT, Optional, default=default value: null]`
`Integer` **`copy`** `[INPUT, Optional, default=default value: 0]`
`QualityContext` **`qualityContext`** `[INOUT, Optional, default=default value: 0]`
`Ramps` **`outramp`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Ramps inRamp [INPUT, MANDATORY, default=NO default value]`

input Frames object

`Ramps rawRamp [INPUT, Optional, default=default value: null]`

input raw Ramps object

`PacsCal calTree [INPUT, Optional, default=default value: null]`

access to PACS calibration tree

RampSatLimits rampSatLimits [INPUT, Optional, default=default value: null]
direct input of cal file rampSatLimits

Integer copy [INPUT, Optional, default=default value: 0]
indicates if the new Frames should be copied and returned (copy =1) or if the old Frames is changed (copy = 0)

QualityContext qualityContext [INOUT, Optional, default=default value: 0]
quality control description

Ramps outramp [OUTPUT, MANDATORY, default=NO default value]
returned Ramps object containing the saturation flags

History

- 2005-08-15 - JS: 1.0, initial version
- 2005-11-15 - JS: 1.1, renamed from flagSaturation to specFlagSaturationRamps!
- 2005-12-05 - JS: 1.2, cal file access added
- 2005-12-09 - JS: 1.3, copy handling implemented
- 2006-03-06 - JS: 1.4, getRampValues() -> getSignal()
- 2006-05-17 - JS: 1.5, minor bug fix
- 2007-08-02 - JS: 1.6, improved version
- 2007-08-23 - JS: 1.7, bug fix, and averaged ramp extrapolation
- 2007-11-26 - JS: 1.8, recoded in java for performance reasons and adapted to new cal framework
- 2009-02-21 - JS: 1.9, again adapt to new cal framework

1.298. SpecMeanDiffChopTask

Full Name:	herschel.pacs.toolboxes.spec.SpecMeanDiffChopTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spec import SpecMeanDiffChopTask

Description

For each ABBA cycle, computes the mean of the unmasked signal pairwise differences.

Different masks can be selected.

API Summary

Jython Syntax
`Frames outFrames = specMeanDiffChop(Frames inFrames)`

Property
`Frames inFrames [INPUT, MANDATORY, default=NO default value]`

API details

Property

`Frames inFrames [INPUT, MANDATORY, default=NO default value]`
Input Frames

History

- 1.0 17/09/2009 CJ Initial version

1.299. specpipe

Full Name:	herschel.pacs.toolboxes.spg.specpipe
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spg import specpipe

History

- 1.0 20071019 JS initial version
- 1.1 20080222 JS adapted according to new cal framework
- 1.2 20080311 JS adapt sequence of modules and add specAssignRaDec
- 1.3 20081017 JS flux cal stubs added

1.300. specProject

Full Name:	herschel.pacs.spg.spec.SpecProjectTask
Alias:	specProject
Type:	Java Task -
Import:	import herschel.pacs.spg.SpecProjectTask
Category:	PACS/Spectrometer/Pipeline

Description

Projects the modules from the spectrometer onto a regular grid on the sky.

This task projects each wavelength of a rebinned cube to a regular RA/Dec grid and adds the WCS RA/Dec coordinates.

Algorithm

1. Scan all RA/Dec values in the inCube and select the unique raster positions. Store for each raster position the frame numbers which match this position, the ra and dec and rotation matrix of the modules (method **selectUniquePositions**).

2. Compute a regular RA/Dec grid which encompasses all the raster positions as computed in the previous step (method **computeGrid**)

3. Loop over all raster positions and do for each position the following:

   a. Compute the weights for projecting the input modules to the output grid. These weights determine which input module(s) the output pixel overlaps and by how much. The results are stored in two 3D arrays: one containing the overlapping modules for each output RA/Dec, and one with their corresponding weights.

   b. Compute for each frame at this position in the cube the output fluxes on the regular grid. This is done by adding up for each pixel the fluxes of the contributing modules multiplied by their overlap weights.

4. Add up the projected images from different raster positions and normalize by dividing with the sum of the weights of all positions.

5. Write the resulting projection to the output cube (exact format needs to be determined).

API Summary

Jython Syntax
`projectedCube = specProject(rebinnedCube,...)`

Properties
`PacsCube | PacsRebinnedCube | ListContext` **`inCube`** `[INPUT, MANDATORY, default=no default value]`
`double` **`outputPixelsize`** `[INPUT, OPTIONAL, default=3.0]`
`boolean` **`use_mindist`** `[INPUT, OPTIONAL, default=false]`
`boolean` **`norm_flux`** `[INPUT, OPTIONAL, default=true]`
`double` **`threshold`** `[INPUT, OPTIONAL, default=2.0]`

Properties
boolean **filter_nans** [INPUT, OPTIONAL, default=false]
boolean **debug** [INPUT, OPTIONAL, default=false]
boolean **interactive** [INPUT, OPTIONAL, default=false]
QualityContext **qualityContext** [INPUT, None, default=no default value]

API details

Properties

PacsCube | PacsRebinnedCube | ListContext inCube [INPUT, MANDATORY, default=no default value]

input cube. If a PacsCube is provided then it is assumed that the grating was not moving and a SimpleImage will be produced.

double outputPixelsize [INPUT, OPTIONAL, default=3.0]

output pixel size (arcsec)

boolean use_mindist [INPUT, OPTIONAL, default=false]

use minimum distance instead of average distance between the spaxels when computing the spaxel size

boolean norm_flux [INPUT, OPTIONAL, default=true]

Divide by exposure map for flux normalization. **EXPERT OPTION. Leave on default value unless you know what you are doing!**

double threshold [INPUT, OPTIONAL, default=2.0]

Only used when PacsCube is provided as input. The minumum jump (in arcsec) which triggers a raster position change

boolean filter_nans [INPUT, OPTIONAL, default=false]

In case this parameter is true then all frames with one or more NaN values will be discarded. USE WITH CAUTION!

boolean debug [INPUT, OPTIONAL, default=false]

Debug mode? (creates extra datasets in output product)

boolean interactive [INPUT, OPTIONAL, default=false]

Interactive mode? (produces several plots during execution)

QualityContext qualityContext [INPUT, None, default=no default value]

Never set this in IA mode. Only used for SPG.

History

- 1.40 2009/01/16 jdejong moved computeGrid out of cube loop
- 1.39 2009/01/16 jdejong Some small updates for handling multiple cubes

- 1.33.2.3 2008/12/10 jdejong Rolled back change for SPR 1177
- 1.33.2.2 2008/12/09 ewieprec SPR 1177
- 1.33.2.1 2008/12/09 ewieprec SPR 1177
- 1.38 2008/12/08 jdejong SPR 1201: Converted threshold for detecting raster positions to degrees
- 1.37 2008/12/05 jdejong SPR 1177: fixed typo in getting the cube dimensions for the exposure map
- 1.36 2008/12/02 jdejong added extra line to convert NaN values to zero
- 1.35 2008/12/01 jdejong SPR 1201: adapted specProject to new FOV orientation
- 1.34 2008/12/01 jdejong SPR 1201: adapted specProject to new FOV orientation
- 1.33 2008/11/12 jdejong branches: 1.33.2;
- 1.32 2008/10/30 jdejong adopted to work with a Context of rebinned cubes
- 1.31 2008/10/28 jdejong added new convenience methods
- 1.30 2008/10/24 jdejong removed the usage of the deprecated PacsProjectedCube class. Now only SimpleCube is used.
- 1.29 2008/10/21 jdejong Fixed handling of NaN values. Some rebinned data contained too many NaNs. Now an average RA/Dec over all non-NaN values is taken
- 1.28 2008/10/21 jdejong Returns now a SimpleCube as result. The PacsProjectedCube class can be used as wrapper for the convenience methods. Also, reintroduced flux normalization (can be turned of with the norm_flux parameter).
- 1.27 2008/10/17 jdejong removed dividing by weight matrix and improved error checks
- 1.26 2008/10/15 jdejong Adapted for working with rebinned cube and fixed WCS
- 1.25 2008/10/01 jdejong added an optional computation of a pseudo flatfield from the background
- 1.24 2008/09/29 jdejong nothing important changed
- 1.23 2008/09/17 jdejong higher threshold for distinguishing raster positions (now 0.1 arcsec)
- 1.22 2008/09/04 jdejong Added weight diagnostic
- 1.21 2008/07/28 jdejong Added NaN check
- 1.20 2008/07/23 jdejong Added some log messages
- 1.19 2008/07/10 jdejong removed some imports
- 1.18 2008/07/10 jdejong fixed to handle now 2D and 3D RA/Dec arrays
- 1.17 2008/07/07 jdejong don't know what changed
- 1.16 2008/06/23 jdejong fixed product generation bugs
- 1.15 2008/06/12 jdejong Changed PacsProjectedCube product to SimpleCube based product.
- 1.14 2008-05-16 jdejong Added RA/Dec of raster positions to product
- 1.13 2008-05-14 jdejong Fixed index order error in some get calls,
- introduced rectangular modules and reorganized

- the data objects for the modules
- 1.12 2008-05-13 jdejong Added printstacktrace call for diagnostics
- 1.11 2008-05-09 jdejong Fixed flux conservation for overlapping raster positions
- 1.10 2008-05-08 jdejong Added weight calibration and much more diagnostic
- data to the product
- 1.9 2008-04-28 jdejong Added PacsProjectedCube product and fixed bug
- concerning selecting image from scan position
- 1.8 2008-04-22 jdejong reorganized some imports
- 1.7 2008-04-18 jdejong Finalized projecting all scan positions on a single grid
- which emcompasses the whole scan range
- 1.6 2008-04-17 jdejong Implemented weighting computation and project method
- 1.5 2008-04-16 jdejong adapted for using PacsCube and implemented up to
- weight function for single grid element
- 1.4 2008-04-16 jdejong intermediate fallback version
- 1.3 2008-04-15 jdejong Implemented corner computation under assumption of
- square module pixels
- 1.2 2008-04-08 jdejong Outcommented mosaic related steps
- 1.1 2008-04-03 jdejong Initial version (does not work yet!!!!)
-

1.301. SpecRespCalTask

Full Name:	herschel.pacs.spg.spec.SpecRespCalTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import SpecRespCalTask

Description

This task divides the signal by the responsivity.

Additionally it propagates the error from the response to the Noise dataset.

As response input it takes either the SpecResponseDriftProduct, the SpecCsResponseAndDarkProduct or the NominalResponse from the corresponding calibration file.

API Summary

Jython Syntax
`Frames outFrame = specRespCal(Frames inFrame[, SpecResponseDriftProduct responseDrift][, SpecCsResponseAndDarkProduct][,PacsCal calTree][,NominalResponse][,int copy])`

Properties
`Frames` **`inFrame`** `[INPUT, MANDATORY, default=NO default value]`
`SpecResponseDriftProduct` **`responseDrift`** `[INPUT, Optional, default=None]`
`SpecCsResponseAndDarkProduct` **`csResponseAndDark`** `[INPUT, Optional, default=None]`
`PacsCal` **`calTree`** `[INPUT, Optional, default=None]`
`NominalResponse` **`nominalResponse`** `[INPUT, Optional, default=None]`
`Integer` **`copy`** `[INPUT, Optional, default=0]`
`Frames` **`outFrame`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
input Frames object

`SpecResponseDriftProduct responseDrift [INPUT, Optional, default=None]`
input product from specFitSignalDrift

`SpecCsResponseAndDarkProduct csResponseAndDark [INPUT, Optional, default=None]`
input product from specDiffCs

PacsCal calTree [INPUT, Optional, default=None]
input calibration tree

NominalResponse nominalResponse [INPUT, Optional, default=None]
input cal file

Integer copy [INPUT, Optional, default=0]
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

Frames outFrame [OUTPUT, MANDATORY, default=NO default value]
returned dark subtracted Frames object

History

- 2008-10-17 - JS: 0.1, initial stub
- 2009-01-28 - JS: 0.2, draft for wavelength switching
- 2009-04-28 - JS: 1.0, first version for everything

1.302. SpecSubtractDarkTask

Full Name:	herschel.pacs.spg.spec.SpecSubtractDarkTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import SpecSubtractDarkTask

Description

This task subtracts either the dark determined from the cal block or the cal file

DarkCurrent. Additionally, it selects the TRG and SWITCH indices from the BlockTable to sort out the calibration block and slews etc.

API Summary

Jython Syntax
`Frames outFrame = specSubtractDark(Frames inFrame, SpecCsResponseAndDarkProduct csResponseAndDark [, PacsCal calTree] [, DarkCurrent darkCurrent])`

Properties
`Frames` **`inFrame`** `[INPUT, MANDATORY, default=NO default value]`
`SpecCsResponseAndDarkProduct` **`csResponseAndDark`** `[INPUT, MANDATORY, default=NO default value]`
`PacsCal` **`calTree`** `[INPUT, Optional, default=default value: null]`
`DarkCurrent` **`darkCurrent`** `[INPUT, Optional, default=default value: null]`
`Frames` **`outFrame`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Frames inFrame [INPUT, MANDATORY, default=NO default value]`
input Frames object

`SpecCsResponseAndDarkProduct csResponseAndDark [INPUT, MANDATORY, default=NO default value]`
input product from specDiffCs

`PacsCal calTree [INPUT, Optional, default=default value: null]`
calibration tree

`DarkCurrent darkCurrent [INPUT, Optional, default=default value: null]`
dark current cal file

Frames outFrame [OUTPUT, MANDATORY, default=NO default value]
returned dark subtracted Frames object

History

- `2008-10-17 - JS`: 0.1 initial stub
- `2009-01-28 - JS`: 0.2 draft for wavelength switching

1.303. SpectrumExtractionTask

Full Name:	herschel.pacs.toolboxes.spg.SpectrumExtractionTask
Type:	Java Task -
Import:	from herschel.pacs.toolboxes.spg import SpectrumExtractionTask

Description

A Task to extract the spectrum of a point source from a PACS cube.

A Task to extract the spectrum of a point source from a PACS cube. In the current version, no corrections have been done for the skew, wave shift and flux normalization due to the mispointing (i.e. the point source in not nicely centered in the middle of the central spaxel).

API Summary

Properties
PacsCube **cube** [INPUT, MANDATORY, default=No default value]
SimpleImage **psf** [INPUT, MANDATORY, default=No default value]
Boolean **oversample** [INPUT, OPTIONAL, default=Default value : true]
double **upsample** [INPUT, OPTIONAL, default=Default value : 3.0]
PacsCal **calTree** [INPUT, OPTIONAL, default=Default value : null]
OPTIONAL **specProperties** [SpecProperties, Default value : null, default=no default value]
Spectrum1d **average** [OUTPUT, MANDATORY, default=No default value]
Spectrum1d **median** [OUTPUT, MANDATORY, default=No default value]
Spectrum1d **sigmaClipped** [OUTPUT, MANDATORY, default=No default value]

API details

Properties

PacsCube cube [INPUT, MANDATORY, default=No default value]

The input PACS cube.

SimpleImage psf [INPUT, MANDATORY, default=No default value]

The input point spread function.

Boolean oversample [INPUT, OPTIONAL, default=Default value : true]

Indicates whether to oversample.

double upsample [INPUT, OPTIONAL, default=Default value : 3.0]

The upsample factor.

PacsCal calTree [INPUT, OPTIONAL, default=Default value : null]

The PACS calibration tree.

OPTIONAL specProperties [SpecProperties, Default value : null, default=no default value]

The PACS spectrometer properties.

Spectrum1d average [OUTPUT, MANDATORY, default=No default value]

The extracted average spectrum.

Spectrum1d median [OUTPUT, MANDATORY, default=No default value]

The extracted median spectrum.

Spectrum1d sigmaClipped [OUTPUT, MANDATORY, default=No default value]

The extracted sigma-clipped spectrum.

1.304. SpecWaveRebinTask

Full Name:	herschel.pacs.spg.spec.SpecWaveRebinTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import SpecWaveRebinTask

Description

Help ...

API Summary

Properties
PacsCube **pacsCube** [INPUT, MANDATORY, default=NO default value]
WavelengthGrid **waveLengthGrid** [INPUT, MANDATORY, default=NO default value]
Boolean **useWeightedMean** [INPUT, OPTIONAL, default=default value: false]
PacsRebinnedCube **PacsRebinnedCube** [OUTPUT, MANDATORY, default=NO default value]

API details

Properties

PacsCube pacsCube [INPUT, MANDATORY, default=NO default value]

input PacsCube object

WavelengthGrid waveLengthGrid [INPUT, MANDATORY, default=NO default value]

input WavelengthGrid object for the re-grid

Boolean useWeightedMean [INPUT, OPTIONAL, default=default value: false]

If true, use weighted mean for flux calibration and calculate uncertainty for noise propagation. Define weighted mean: Let "x" be a vector of n numbers for input and "dx" be a vector of "n" numbers that represent the uncertainties of "x" in the same units. Let "x[i]" be the i'st element of a vector "x". Let sum() be the sum over all indices i from 1 to n. The weighted mean wmean is then: wmean = sum(x[i]/(dx[i]^2)) / sum(1/(dx[i]^2)) Define uncertainty (noise): The uncertainty of the weighted mean is then: ewmean = sqrt(sum((x[i]-wmean)^2/(dx[i]^2)) / sum(1/(dx[i]^2)))

PacsRebinnedCube PacsRebinnedCube [OUTPUT, MANDATORY, default=NO default value]

returned cube with number of wavelength bins x 5 x 5 browse quality cube.

1.305. spg_pacsphotsmallextended

Full Name:	herschel.pacs.toolboxes.spg.spg_pacsphotsmallextended
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spg import spg_pacsphotsmallextended

History

- 1.0 20080416 EkW initial version - basing on pacsphotsmallextended.py
- 2.0 20080417 EkW re-write to match into pipeline architecture

1.306. SubtractOpenFramesTask

Full Name:	herschel.pacs.toolboxes.spec.SubtractOpenFramesTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spec import SubtractOpenFramesTask

Description

subtracts the signals of the open channel pixels from the signals of each corresponding pixel

this function subtracts the signals of the open channel pixels from the signals of each pixel of the corresponding detector module.

API Summary

Jython Syntax
`Frames frame = subtractOpenFrames(Frames inFrames [, int copy = <copy>])`

Properties
`Frames inFrames [INPUT, MANDATORY, default=NO default value]`
`Integer copy [INPUT, Optional, default=default value: 0]`
`Frames outFrame [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Frames inFrames [INPUT, MANDATORY, default=NO default value]`
input Frames object

`Integer copy [INPUT, Optional, default=default value: 0]`
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

`Frames outFrame [OUTPUT, MANDATORY, default=NO default value]`
returned frame with open channel signals subtracted

History

- 1.0 22-Apr-2005 JS initial version
- 1.1 05-Aug-2005 JS header adopted to jtags syntax
- 1.2 15-Sep-2005 JS Frames import was missing
- 1.3 14-Dez-2005 JS conversion to task
- 1.4 22-Mar-2006 JS debugged and performance improved version
- 1.5 29-May-2006 JS performance improved version

1.307. SubtractOpenRampsTask

Full Name:	herschel.pacs.toolboxes.spec.SubtractOpenRampsTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spec import SubtractOpenRampsTask

Description

subtracts the ramps of the open channel pixels from the Ramps of each corresponding pixel

this function subtracts the ramps of the open channel pixels from the ramps of each pixel of the corresponding detector module.

API Summary

Jython Syntax
`Ramps ramp = subtractOpenRamps(Ramps inRamps [, int copy = <copy>])`

Properties
`Ramps inRamps [INPUT, MANDATORY, default=NO default value]`
`Integer copy [INPUT, Optional, default=default value: 0]`
`Ramps ramp [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

Ramps inRamps [INPUT, MANDATORY, default=NO default value]
input Ramps object

Integer copy [INPUT, Optional, default=default value: 0]
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

Ramps ramp [OUTPUT, MANDATORY, default=NO default value]
returned Ramps object with open channel ramps subtracted

History

- 1.0 09-Mai-2005 JS initial version
- 1.1 20050524 JS imports deleted
- 1.2 20050620 JS minor changes
- 1.3 20050805 JS header adopted to jtags syntax
- 1.4 20050915 JS Ramps import was missing
- 1.5 20051214 JS conversion to task

- 1.6 20060306 JS getRampValues() -> getSignal()
- 1.7 20060322 JS debugged and performance improved version
- 1.8 20060529 JS performance improved version
- 1.9 20070822 JS improved version

1.308. UniqTask

Full Name:	herschel.pacs.toolboxes.numeric.UniqTask
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import UniqTask

Description

Return the subscripts of the unique elements in an array

Note that repeated elements must be adjacent in order to be found. This routine is intended to be used with the SORT function.

API Summary

Jython Syntax
`idx = UNIQ(array[, first=0])`

Properties
`Numeric1dData array [INPUT, MANDATORY, default=NO default value]`
`Boolean first [INPUT, OPTIONAL, default=0]`
`Int1d idx [OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Numeric1dData array [INPUT, MANDATORY, default=NO default value]`
sorted array of numbers

`Boolean first [INPUT, OPTIONAL, default=0]`
return index of FIRST occurence for duplicates (default is LAST occurence)

`Int1d idx [OUTPUT, MANDATORY, default=NO default value]`
an array of indices into the original array

History

- 28-Sep-2004 RH Initial version of this task
- 01-Oct-2004 RH renamed uniq to UNIQ
- 09-Mar-2005 RH import of SHIFT removed since it is automatically imported
- 04-Aug-2005 JS header adopted to jtags concept
- 07-Nov-2005 JS clean up of imports

1.309. utils_reverse

Full Name:	herschel.pacs.toolboxes.numeric.utils_reverse
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import utils_reverse

Description

Utilities for manipulating numeric arrays.

This utils module contains a number of functions to manipulate numerical arrays. The reverse4d() function reverses a four-dimensional array along the given dimension which is the second dimension by default. The reverse5d() function reverses a five-dimensional array in the given dimension. The dimension can be specified with the keyword 'dim' and defaults to the fifth dimension i.e. dim=4.

Miscellaneous

- no proper type and rank checking done in all functions

History

- `2005-05-27 - SG`: - First version
- `2005-08-16 - SG`: - put it into a toolbox, made type independent and added docstrings
- `2005-08-18 - SG`: - header adopted to jtags syntax
- `2005-11-07 - JS`: - clean up of imports

1.310. utils_rotate

Full Name:	herschel.pacs.toolboxes.numeric.utils_rotate
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import utils_rotate

Description

Utilities for manipulating numeric arrays.

This utils module contains a number of functions to manipulate numerical arrays. The rotate4d() function rotates a four-dimensional array along the given dimension (dim=1, dim=2, dim=3). The rotation is done clockwise by default unless the 'dir' keyword is set to -1. The dimension can be specified with the keyword 'dim' i.e. dim=3. The rotate5d() function rotates a five-dimensional array along the given dimension (dim=2, dim=3, dim=4). The rotation is done clockwise by default unless the 'dir' keyword is set to -1. The dimension can be specified with the keyword 'dim' i.e. dim=3.

Miscellaneous

- no proper type and rank checking done in all functions

History

- 2005-06-30 SG- First version
- `2005-08-16 - SG`: - put it into a toolbox, made type independent and added docstrings)
- `2005-08-18 - SG`: - header adopted to jtags syntax
- `2005-11-07 - JS`: - clean up of imports

1.311. utils_transpose

Full Name:	herschel.pacs.toolboxes.numeric.utils_transpose
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import utils_transpose

Description

Utilities for manipulating numeric arrays.

This utils module contains a number of functions to manipulate numerical arrays. The transpose4d() function transposes a fourth-dimensional array along the given dimension (dim=1, dim=2, dim=3). The dimension can be specified with the keyword 'dim' i.e. dim=3. The transpose5d() function transposes a fifth-dimensional array along the given dimension (dim=2, dim=3, dim=4). The dimension can be specified with the keyword 'dim' and defaults to the five dimension i.e. dim=4.

Miscellaneous

- no proper type and rank checking done in all functions

History

- `2005-06-06 - SG`: - First version
- `2005-08-16 - SG`: - put it into a toolbox, made type independent and added docstrings)
- `2005-08-18 - JS`: - header adopted to jtags syntax

1.312. utils_void_cube

Full Name:	herschel.pacs.toolboxes.numeric.utils_void_cube
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import utils_void_cube

Description

Utilities for manipulating numeric arrays.

This utils module contains a number of functions to manipulate numerical arrays. Insert a void columns/rows or plane in ArrayData cube the original cube voidPositions the indices in the original cube after which to insert the row(plane-xz)/column(plane-xy) or plane-yz voidValue the value to be inserted (defalt = 0)

Miscellaneous

- no proper type and rank checking done in all functions

History

- `2006-02-24 - SG`: - First version
- `2006-06-21 - SG`: - new version
- `2006-07-27 - SG`: - put it into a toolbox, made type independent and added docstrings and header adopted to jtags syntax

1.313. utils_void_fifth

Full Name:	herschel.pacs.toolboxes.numeric.utils_void_fifth
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import utils_void_fifth

Description

Utilities for manipulating numeric arrays.

This utils module contains a number of functions to manipulate numerical arrays. Insert a void columns/rows or plane in ArrayData fifth the original Fifth voidPositions the indices in the original fifthafter which to insert the column/row or plane voidValue the value to be inserted (defalt = 0)

Miscellaneous

- no proper type and rank checking done in all functions

History

- `2006-06-12 - SG`: - First version
- `2006-07-17 - SG`: - New version
- `2006-07-27 - SG`: - put it into a toolbox, made type independent and added docstrings and header adopted to jtags syntax

1.314. utils_void_fourth

Full Name:	herschel.pacs.toolboxes.numeric.utils_void_fourth
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import utils_void_fourth

Description

Utilities for manipulating numeric arrays.

This utils module contains a number of functions to manipulate numerical arrays. Insert a void columns/rows or planes in ArrayData Fourth the original Fourth voidPositions the indices in the original Fourth after which to insert the column/row or plane voidValue the value to be inserted (defalt = 0)

Miscellaneous

- no proper type and rank checking done in all functions

History

- `2006-06-9 - SG`: - First version
- `2006-06-22 - SG`: - New version
- `2006-07-27 - SG`: - put it into a toolbox, made type independent and added docstrings and header adopted to jtags syntax

1.315. utils

Full Name:	herschel.pacs.toolboxes.binstruct.utils
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.binstruct import utils

Description

Utilities for investigating and manipulating packets and packet sequences.

This utils module contains a number of functions to investigate and manipulate packets and packet sequences. readTm: Read a tm-file into a PacketSequence. Without arguments this function shows a file selector box to select the tm-file. The filenames are filtered on the pattern '.tm'. If the property 'pacs.tm.datapath' exists, this path is used as a starting point for the file selector box. dumpPackets: For each packet the following columns are printed: # number of the packet in the sequence V version number T source type F data flag APID application id SF segmentation flags S_CNT source sequence count Length packet length PV pus version Type packet type SubT packet Sub Type Time packet time TM Data source data (first 8 words) ip = switchToHpsdbAsciiServices() ip = switchToPibServices() ip = switchToMibAsciiServices() functions to switch between different formats of the MIB definitions in binstruct setTmVersions: re-initialise binstruct for the given telemetry version showTmVersions: print the names of the available TmVersions tables identifyUndefinedApids: returns the apids of the unidentified packets in the given packet sequence

API Summary

Jython Syntax
`PacketSequence seq = readtm([String filename = <filename>])`
`dumppackets(PacketSequence seq[, int m=0, int n=10])`

`ip = switchToHpsdbAsciiServices()`
`ip = switchToPibServices()`
`ip = switchToMibAsciiServices()`

`ip = setTmVersions(name)` `showTmVersions()`
`print identifyUndefinedApids(seq)`

Properties
`String filename [INPUT, OPTIONAL, default=NO default value]`
`PacketSequence seq [OUTPUT, MANDATORY, default=NO default value]`
`PacketSequence seq [INPUT, MANDATORY, default=NO default value]`
`int m [INPUT, OPTIONAL, default=0]`
`int n [INPUT, OPTIONAL, default=10]`
`String name [INPUT, OPTIONAL, default=default]`

API details

Properties

String filename [INPUT, OPTIONAL, default=NO default value]
filename string

PacketSequence seq [OUTPUT, MANDATORY, default=NO default value]

returned PacketSequence

PacketSequence seq [INPUT, MANDATORY, default=NO default value]

int m [INPUT, OPTIONAL, default=0]

m offset for the first packet to dump

int n [INPUT, OPTIONAL, default=10]

n number of packets to dump (default=10)

String name [INPUT, OPTIONAL, default=default]

name of the TmVersions table (the part after the dash (-) without the extention)

History

- 2004-09-28 - RH: - Initial version of this toolbox based on an idea of Dieter Lutz
- 2004-10-12 - RH: - added dumppackets function
- 2005-07-28 - JS: - header converted to jtags framework
- 2005-11-07 - JS: - clean up of imports
- 2009-03-11 - RH: - added setTmVersions(), showTmVersions() and identifyUndefinedApids()
- 2009-06-01 - RH: - added switchTo...Services() methods

1.316. utils

Full Name:	herschel.pacs.toolboxes.numeric.utils
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.numeric import utils

Description

Utilities for manipulating numeric arrays.

This utils module contains a number of functions to manipulate numerical arrays. The rotate2d() function rotates a two-dimensional array. The rotation is done clockwise by default unless the 'dir' keyword is set to -1. The rotate3d() function rotates a three-dimensional array (a cube) along the given dimension. The rotation is done clockwise by default unless the 'dir' keyword is set to -1. The dimension can be specified with the keyword 'dim' and defaults to the third dimension i.e. dim=2 (depth of the cube). The transpose2d() function transposes a two-dimensional array (matrix transposition). The transpose3d() function transposes a three-dimensional array along the given dimension. The dimension can be specified with the keyword 'dim' and defaults to the third dimension i.e. dim=2 (depth of the cube). The reverse2d() function reverses a two-dimensional array along the given dimension which is the second dimension by default. The reverse3d() function reverses a three-dimensional array in the given dimension. The dimension can be specified with the keyword 'dim' and defaults to the third dimension i.e. dim=2 (depth of the cube).

Miscellaneous

- no proper type and rank checking done in all functions

History

- `2004-09-23 - BV`: - First version
- `2004-09-28 - RH`: - put it into a toolbox, made type independent and added docstrings
- `2004-09-30 - RH`: - all functions and combinations provided and optimized
- `2005-08-04 - JS`: - header adopted to jtags syntax
- `2005-11-07 - JS`: - clean up of imports

1.317. visibilityToolPacs

Full Name:	herschel.pacs.toolboxes.visibility.visibilityToolPacs
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.visibility import visibilityToolPacs

Description

Jython script to calculate visibility given RA, DEC and constraints

Contents: isVisible() printPosAngle() runVisibility() checkVisibility() reformOutput() getBodyRaDec() runVisibilityOnTargetList()

History

- `2007-12-20 - RV`: - visibilityToolPacs.py created for PCSS based on IV tools
- `2008-01-08 - BV`: - small corrections for insertion in PCSS

1.318. wave2GratPos

Full Name:	herschel.pacs.toolboxes.spec.wave2GratPos
Type:	Jython Task -
Import:	from herschel.pacs.toolboxes.spec import wave2GratPos

Description

Calculate the grating position corresponding to a wavelength

API Summary

Jython Syntax
`double gratPos = wave2GratPos(double wave, String band, int pix, int mod`
`[, PacsCal calTree][, LittrowPolynomes littrowPolynomes]`
`[, model = "FM"][, time=Date()])`
`Double1d gratPos = wave2gratPos(Double1d wave, String band, int pix, int mod`
`[, PacsCal calTree][, LittrowPolynomes littrowPolynomes]`
`[, model = "FM"][, time=Date()])`

Properties
`Double1d` **`wave`** `[INPUT, MANDATORY, default=NO default value]`
`String` **`band`** `[INPUT, MANDATORY, default=NO default value]`
`int` **`pix`** `[INPUT, MANDATORY, default=NO default value]`
`int` **`mod`** `[INPUT, MANDATORY, default=NO default value]`
`PacsCal` **`calTree`** `[INPUT, OPTIONAL, default=None]`
`LittrowPolynomes` **`littrowPolynomes`** `[INPUT, OPTIONAL, default=None]`
`String` **`model`** `[INPUT, Optional, default=None]`
`FineTime` **`time`** `[INPUT, Optional, default=None]`
`Double1d` **`gratPos`** `[OUTPUT, MANDATORY, default=NO default value]`

API details

Properties

`Double1d wave [INPUT, MANDATORY, default=NO default value]`
Computed wavelength in um

`String band [INPUT, MANDATORY, default=NO default value]`
Spectral band: B3A : Blue short wave channel : filter A, order=3, 50-70um B2A : Blue short wave channel : filter A, order=2, 50-70um B2B : Blue long wave channel : filter B, order=2, 70-100um R1 : Red channel : order=1, 100-200um

`int pix [INPUT, MANDATORY, default=NO default value]`
Pixel number within a module (1,2,3 .., 16)

int mod [INPUT, MANDATORY, default=NO default value]

Module number (0,1,2,3 .., 24)

PacsCal calTree [INPUT, OPTIONAL, default=None]

cal file access

LittrowPolynomes littrowPolynomes [INPUT, OPTIONAL, default=None]

Specific version of LittrowPolynomes calibration table to use

String model [INPUT, Optional, default=None]

model name FM, FS or QM

FineTime time [INPUT, Optional, default=None]

time of measurement

Double1d gratPos [OUTPUT, MANDATORY, default=NO default value]

computed grating positions

History

- 1.0 01-02-2006 - BV - First version
- 1.1 18-03-2007 - BV - Changed default calibration table to FM_1_0
- 1.2 22-02-2008 - JS - adapted to new cal framework
- 2.0 12-03-2008 - BV - Changed for use of LittrowPolynomes
- 2.1 20-03-2008 - BV - changed calfile access selection cfr common approach

1.319. WaveCalcTask

Full Name:	herschel.pacs.spg.spec.WaveCalcTask
Type:	Jython Task -
Import:	from herschel.pacs.spg.spec import WaveCalcTask

Description

Calculates the wavelength for every pixel in every frame

The wavelength seen by a PACS pixel can be calculated by translating the position of the diffraction grating to the angle between the pixel and the grating surface. The PACS grating is used in littrow configuration (i.e. infalling angle = outgoing angle), hence this processing step evaluates the littrow equation for this angle determined from the grating position. This relation is not the same for every pixel and is affected by distortions. Therefore, the relation littrow angle - grating position is described by a fifth order polynome for every pixel in the calibration table cal.spectrometer.littrowPolynomes. This calibration table lists also a number of other parameters of the littrow equation that are common between the pixels (grating steps per degree, grating constant, etc)

This processing step evaluates for every pixel in every frame the littrow equation using the parameters in the calibration table. Depending on the detector band used, and the position of the order selection filter wheel in the blue chain, the visible order is determined and the corresponding wavelength is added to the *wave* dataset of the frames. The status field *band* is populated with the name of the spectral band seen in the frame:

- R1 : order 1, red detector (100-220 um)
- B2B : order 2, blue detector (70-100 um)
- B2A : order 2, blue detector (50-70 um)
- B3A : order 3, blue detector (50-70 um)

API Summary

Jython Syntax
`Frames outFrames = waveCalc(Frames inFrames[,calTree=calTree] [littrowPolynomes=littrowPolynomes] [, int copy = <copy>])`

Properties
`Frames inFrames [INPUT, MANDATORY, default=NO default value]`
`Frames outFrames [OUTPUT, MANDATORY, default=NO default value]`
`PacsCal calTree [INPUT, OPTIONAL, default=None]`
`LittrowPolynoms littrowPolynomes [INPUT, OPTIONAL, default=None]`
`Integer copy [INPUT, Optional, default=default value: 0]`
`String band [INPUT, OPTIONAL, default=default value : automatic]`

API details

Properties

Frames inFrames [INPUT, MANDATORY, default=NO default value]
Frames input class for wavelength calculation

Frames outFrames [OUTPUT, MANDATORY, default=NO default value]
Depending on copy keyword a copy of the Frames class with converted digits to Volts or an updated Frames class

PacsCal calTree [INPUT, OPTIONAL, default=None]
cal file access

LittrowPolynoms littrowPolynomes [INPUT, OPTIONAL, default=None]
cal file

Integer copy [INPUT, Optional, default=default value: 0]
indicates if the new frames should be copied and returned (copy =1) or if the old frame is changed (copy = 0)

String band [INPUT, OPTIONAL, default=default value : automatic]
You may force to use a certain spectral band R1, B2A, B3A, B2B

See also

- : Wavelength Calibration of the PACS Spectrometer AVM/CQM
- PTD 4.2.1 FGB, Issue 1.0, feb 4,2005

History

- 1.0 07-Mar-2005 EkW Initial version of this Task
- Basing on BVs py script and FGBs Litrow equation
- 1.1 05-Aug-2005 JS Header adopted to jtags syntax
- 1.2 07-Nov-2005 JS Clean up of imports
- 1.3 09-Nov-2005 EkW Usage of calibration file and gratPos2Wave
- 1.4 06-Mar-2006 EkW Unit Handling
- 1.5 08-Mar-2006 EkW Add copy keyword
- 1.6 24-May-2006 EkW len() on Selection to .length()
- 1.7 15-May-2006 EkW Add pfilter keyword
- 2.0 20-Nov-2007 EkW Adopt to new common calibration framework
- 2.1 22-Feb-2008 JS adapt again to new cal framework
- 3.0 12-Mar-2008 BV Use the per-pixel-and-module calibration
- 3.1 23-Apr-2009 BV SPR PACS 1543: pass the calfile instead of leaving this
- to gratPos2Wave
- 3.2 16-Nov-2009 BV Improve URM documentation

1.320. WavelengthGrid

Full Name:	herschel.pacs.cal.spectrometer.WavelengthGrid
Type:	Java Class -
Import:	from herschel.pacs.cal.spectrometer import WavelengthGrid

Description

Wavelength grid for the PACS spectrometer.

Wavelength grid for the PACS spectrometer, for the three grating orders and for different upsamples.

API Summary

Constructors
`WavelengthGrid()`
The construction of a new WavelengthGrid.
`WavelengthGrid(WavelengthGrid original)`
Copy constructor for WavelenghtGrid.

Methods
`importFrom(Product product)`
Imports a WavelengthGrid from a vanilla product.
`WavelengthGrid` **`importFrom`**`(WavelengthGrid grid, Product product)`
Instantation of the wavelength grid calibration data from a vanilla product.
`setWavelength``(Double1d wavelength, int order, int upsample)`
Setting the wavelengths.
`setResolvingPower``(Double1d resolvingPower, int order, int upsample)`
Setting the resolving powers.
`setData``(TableDataset data, int order, int upsample)`
Setting the data.
`Double1d` **`getWavelength`**`(int order, int upsample)`
Returns the wavelengths.
`Double1d` **`getResolvingPower`**`(int order, int upsample)`
Returns the resolving powers.
`TableDataset` **`getData`**`(int order, int upsample)`
Returns the data.

API details

Constructors

`WavelengthGrid()`
The construction of a new WavelengthGrid.

WavelengthGrid()

A new WavelengthGrid is constructed and the meta data is set according to the rules of the calibration framework.

WavelengthGrid(WavelengthGrid original)

Copy constructor for WavelenghtGrid.

Makes a copy of the given WavelengthGrid.

Argument

`WavelengthGrid` **`original`** `[INPUT, MANDATORY]`

Methods

importFrom(Product product)

Imports a WavelengthGrid from a vanilla product.

Imports a WavelengthGrid from a vanilla product, by copying the meta data and the datasets.

Argument

`Product` **`product`** `[INPUT, MANDATORY]`

WavelengthGrid importFrom(WavelengthGrid grid, Product product)

Instantation of the wavelength grid calibration data from a vanilla product.

Instantiates the wavelengths grid calibration data from a vanilla product, by copying the meta data and the datasets.

Arguments

`WavelengthGrid` **`grid`** `[INPUT, MANDATORY]`

`Product` **`product`** `[INPUT, MANDATORY]`

Return

WavelengthGrid

Returns the instantiated wavelength grid calibration data.

setWavelength(Double1d wavelength, int order, int upsample)

Setting the wavelengths.

Sets the wavelengths for the given grating order and number of upsamples for this WavelengthGrid to the given list of wavelengths.

Arguments

`Double1d` **`wavelength`** `[INPUT, MANDATORY]`

`int` **`order`** `[INPUT, MANDATORY]`

`int` **`upsample`** `[INPUT, MANDATORY]`

setResolvingPower(Double1d resolvingPower, int order, int upsample)

Setting the resolving powers.

Sets the resolving powers for the given grating order and number of upsamples for this WavelengthGrid to the given list of wavelengths.

setResolvingPower(Double1d resolvingPower, int order, int upsample)

Arguments

Double1d **resolvingPower** [INPUT, MANDATORY]

int **order** [INPUT, MANDATORY]

int **upsample** [INPUT, MANDATORY]

setData(TableDataset data, int order, int upsample)

Setting the data.

Sets the data for the given grating order and number of upsamples for this WavelengthGrid to the given data.

Arguments

TableDataset **data** [INPUT, MANDATORY]

int **order** [INPUT, MANDATORY]

int **upsample** [INPUT, MANDATORY]

Double1d getWavelength(int order, int upsample)

Returns the wavelengths.

Returns the wavelenghts for the given grating order and number of upsamples for this WavelengthGrid.

Arguments

int **order** [INPUT, MANDATORY]

int **upsample** [INPUT, MANDATORY]

Return

Double1d

Returns the wavelengths for the given grating order and number of upsamples for this WavelengthGrid.

Double1d getResolvingPower(int order, int upsample)

Returns the resolving powers.

Returns the resolving powers for the given grating order and number of upsamples for this WavelengthGrid.

Arguments

int **order** [INPUT, MANDATORY]

int **upsample** [INPUT, MANDATORY]

Return

Double1d

Returns the resolving powers for the given grating order and number of upsamples for this WavelengthGrid.

TableDataset getData(int order, int upsample)

Returns the data.

`TableDataset getData(int order, int upsample)`

Returns the data for the given grating order and number of upsamples for this WavelengthGrid.

Arguments

`int` **`order`** `[INPUT, MANDATORY]`

`int` **`upsample`** `[INPUT, MANDATORY]`

Return

TableDataset

Returns the data for the given grating order and number of upsamples for this WavelengthGrid.

1.321. WavelengthGrid

Full Name:	herschel.pacs.signal.WavelengthGrid
Type:	Java Class -
Import:	from herschel.pacs.signal import WavelengthGrid

Description

See task spg/WavelengthGridTask for how to create and

use WavelengthGrid. Find parameters used in creating this grid: waveGrid = wavelengthGrid(pacsCube) print "Upsample = ", waveGrid.upsample print "Oversample = ", waveGrid.oversample print "Band = ", waveGrid.band print "Order = ", waveGrid.order # To get all meta data: print waveGrid.meta

Example

Example 1: Get start, centre and end values for the wavlength grids:

```
wavelengthGridSet = waveGrid.getWavelengthGridSet()
starts = wavelengthGridSet[WavelengthGrid.STARTS,:]
centres = wavelengthGridSet[WavelengthGrid.CENTRES,:]
ends    = wavelengthGridSet[WavelengthGrid.ENDS,:]
```

API Summary

Properties
PacsCube **pacsCube** [INPUT, MANDATORY, default=NO default value]
double **oversample** [INPUT, OPTIONAL, default=Default = 1.0]
WavelengthGrid **Wavelength** [OUTPUT, MANDATORY, default=NO default value]

API details

Properties

PacsCube pacsCube [INPUT, MANDATORY, default=NO default value]

input PacsCube object

double oversample [INPUT, OPTIONAL, default=Default = 1.0]

WavelengthGrid Wavelength [OUTPUT, MANDATORY, default=NO default value]

returned reference to Product containing wavelength information.

1.322. WavelengthGridTask

Full Name:	herschel.pacs.spg.spec.WavelengthGridTask
Type:	Java Task -
Import:	from herschel.pacs.spg.spec import WavelengthGridTask

Description

Help ...

API Summary

Properties
PacsCube **pacsCube** [INPUT, MANDATORY, default=NO default value]
double **oversample** [INPUT, MANDATORY, default=default value = 1.0]
double **upsample** [INPUT, MANDATORY, default=default value = 4.0]
Double1d **wavelengthGrid** [OUTPUT, MANDATORY, default=NO default value]

API details

Properties

PacsCube pacsCube [INPUT, MANDATORY, default=NO default value]

input PacsCube object

double oversample [INPUT, MANDATORY, default=default value = 1.0]

input oversample is used to increase the number of wavelength bins by this formula: bins * oversample, where bins is the count of wavelength bins based on the theoretical resolution.

double upsample [INPUT, MANDATORY, default=default value = 4.0]

input upsample specifies how many shifts per wavelength bin to make while re-binning. The default value of 4.0 used with an oversample of 1.0 has been shown to improve improve the detection of faint lines. Each bin is sampled "upsample" times, shifting upwards in wavelength by 1/upsample. For example, upsample value of 2 means sample, shift by binwidth / 2, and sample again, and since both samples are the width of the original wavelength bin, the second sample overlaps the next bin.

Double1d wavelengthGrid [OUTPUT, MANDATORY, default=NO default value]

returned Double1d containing wavelengths for re-binning. The values in the wavelength grid are the centre values of each bin. Note: use wavelengthGrid.getWaveGridSet() to return a Double2d containing the edges and centre of each bin.

1.323. Wavelet

Full Name:	herschel.pacs.spg.phot.deglitching.WTMML.wavelet.Wavelet
Type:	Java Class -
Import:	from herschel.pacs.spg.phot.deglitching.WTMML.wavelet import Wavelet
Category:	class

History

- 24/04/2006 : first version
- 22/02/2007 : admissibility factor added in variables definition
- 08/03/2007 : compute only c_psi between scales 0 and f_x_max

1.324. waveSwitchL1

Full Name:	herschel.pacs.spg.pipeline.ipipe.spec.waveSwitchL1
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.ipipe.spec import waveSwitchL1

History

- 1.0 20090318 JS initial version
- 1.1 20090624 JS updated
- 1.2 20090716 JS introduce slicing

1.325. waveSwitchL2

Full Name:	herschel.pacs.spg.pipeline.ipipe.spec.waveSwitchL2
Type:	Jython Task -
Import:	from herschel.pacs.spg.pipeline.ipipe.spec import waveSwitchL2

History

- 1.0 20090318 JS initial version
- 1.1 20090624 JS updated
- 1.2 20090716 JS introduce slicing

1.326. Wcs4mapTask

Full Name:	herschel.pacs.spg.phot.Wcs4mapTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import Wcs4mapTask

Description

This task calculates a Wcs from the frames object. The Wcs is designed in a way, that all signal values

of the frames are projected into the corresponding map.

The Wcs may be modified and used as input for PhotProjectTask in order to custumize the design of the final map.

Parameters:

Example

Example 1: from hipe:

```
from herschel.pacs.spg import Wcs4mapTask
wcs4map = Wcs4mapTask()
wcs = wcs4map(frames)
<br>
wcs = wcs4map(frames, optimizeOrientation = True)
```

API Summary

Properties
MANDATORY **inframes: INPUT** [Frames, no default value., default=no default value]
OPTIONAL **outputPixelsize: INPUT** [DOUBLE, default is the same value as the inframes pixelsize (3.2 arcsecs for the blue photometer, default=6.4 for red).]
OPTIONAL **optimizeOrientation: INPUT** [Boolean, default is False., default=no default value]
OPTIONAL **mapcoordinates: INPUT** [Double1d, no default value., default=no default value]
OPTIONAL **wcs:OUTPUT** [Wcs, no default value., default=no default value]

API details

Properties

MANDATORY inframes: INPUT [Frames, no default value., default=no default value]

the input frames object

OPTIONAL outputPixelsize: INPUT [DOUBLE, default is the same value as the inframes pixelsize (3.2 arcsecs for the blue photometer, default=6.4 for red).]

the size of a pixel in the output dataset in arcseconds. Default is the same size as the input (6.4 arcsecs for the red and 3.2 arcsecs for the blue photometer).

OPTIONAL optimizeOrientation: INPUT [Boolean, default is False., default=no default value]

rotates the map by an angle between minRotation and 89 degrees in a way that the coverage of the outputimage is maximized

as a result, north points no longer upwards.

Possible values False (default): no automatic rotation, True: automatic rotation

* @jparameter minRotation: INPUT, Integer, OPTIONAL, default value is 15. defines the minimum angle for which optimizeOrientation should be applied. Default value is 15 degrees. If optimizeOrientation finds out that the angle is below this value, no rotation will be done. A message is logged instead.

OPTIONAL mapcoordinates: INPUT [Double1d, no default value., default=no default value]

This paramter is deprecated and will be removed.

OPTIONAL wcs:OUTPUT [Wcs, no default value., default=no default value]

The calculated World Coordinate System is the output of this task.

1.327. WTMMLDeglitchingTask

Full Name:	herschel.pacs.spg.phot.WTMMLDeglitchingTask
Type:	Java Task -
Import:	from herschel.pacs.spg.phot import WTMMLDeglitchingTask
Category:	class

API Summary

Jython Syntax
`outFrames = wtmmlDeglitching(inFrames [,copy=copy][,scaleMin=2.0] [,scaleMax=6.0][,hmin=-1.3][,voices=5.][,holderThreshold=-0.6] [,CorrThreshold=0.985][,reconstruction=True]`

Properties
type **inFrames : the input frame object containing signal to analyse** [INPUT, MANDATORY, default=no default value]
type **copy : boolean with the possible values :** [INPUT, MANDATORY, default=no default value]
type **minScale : signal continuous wavelet transform is computed from scaleMin** [INPUT, MANDATORY, default=no default value]
type **maxScale : signal continuous wavelet transform is computed till scaleMax** [INPUT, MANDATORY, default=no default value]
type **voices : voices number by octave** [INPUT, MANDATORY, default=no default value]
type **hmin : the minimal holder value allowed** [INPUT, MANDATORY, default=no default value]
type **holderThreshold : the threshold holder exponent** [must be greater than hmin, MANDATORY, default=no default value]
type **corrThreshold : correlation coefficient threshold (around 1.) is a criteria used** [INPUT, MANDATORY, default=no default value]
type **reconstuction : Boolean: {true|false} = {inFrames is changed | inFrames is not changed}** [INPUT, MANDATORY, default=no default value]

API details

Properties

type inFrames : the input frame object containing signal to analyse [INPUT, MANDATORY, default=no default value]

type copy : boolean with the possible values : [INPUT, MANDATORY, default=no default value]

false (jython: 0) - inFrames will be modified and returned true (jython: 1) - a copy of inFrames will be returned

type minScale : signal continuous wavelet transform is computed from scaleMin [INPUT, MANDATORY, default=no default value]

type maxScale : signal continuous wavelet transform is computed till scaleMax [INPUT, MANDATORY, default=no default value]

type voices : voices number by octave [INPUT, MANDATORY, default=no default value]

type hmin : the minimal holder value allowed [INPUT, MANDATORY, default=no default value]

type holderThreshold : the threshold holder exponent [must be greater than hmin, MANDATORY, default=no default value]

type corrThreshold : correlation coefficient threshold (around 1.) is a criteria used [INPUT, MANDATORY, default=no default value]

to identify an irregularity of the signal

type reconstuction : Boolean: {true|false} = {inFrames is changed | inFrames is not changed} [INPUT, MANDATORY, default=no default value]

Wavelet transfom will be computed from scaleMin to scaleMax. Octave number (nOct) is log(scaleMax)/log (2) (dyadic decomposition) and there are nVoice voices by octave. The scale a of the octave o and the voice v is a = 2^(nOct*(o-1)+v/nVoice) Correlation threshold (close to value 1.) is a criteria used to identify a potentially irregularity of the signal as a possible glitch. holderThreshold gives an upper limit of the acceptable holder exponent found hmin gives a lower limit of the acceptable holder exponent found hmin < holder exponent found < holderThreshold

History

- `2007-06-12 - BM`: : initial version