
**SUBJECT:** **SPIRE Data Processing Pipeline Module Requirements**

**PREPARED BY:** **D.L. Clements**

**DOCUMENT No:** **SPIRE-ICS-DOC-002998**

**ISSUE:** **Draft 1 .3** **Date:** **25th July 2008**

**APPROVED BY:** **Date:**

# Distribution

# Change Record

| ISSUE | DATE | Changes |
|---|---|---|
| 1 | | First Draft |
| 1.1 | | Includes updates from developers after Chris' updated pipeline description |
| 1.2 | | Include further updates |

Changes in 1.1 (TRF, PD):

Removed channel Fringe Correction from Table 1 as it is not part of the baseline delivery.
Modified the following package owner in Table 1 and in the document body:

- Clipping correction changed from Christophe to Dominique/LAM,
- SCAL/Telescope correction changed from Christophe to Dominique/LAM,
- Regrid Onto Regular Sky Grid (Spectral Cube) changed from TBD to Ralph.

**NB:** The numbers in the first column of Table 1 may need to be modified to reflect the changes in this version.

Modified the input, input calibration, output, and functional requirements sections of:

- Correct Time Domain Phase
- Create Interferograms
- Baseline Correction (Spectrometer)
- 2nd Level Deglitching (Spectrometer)
- Apodisation
- Fourier Transform
- Phase Correction
- Average Spectra
- Spectral Response Correction (Spectrometer)

Added section placeholders for:

- Spectral Flux Conversion
- Remove Optical Crosstalk (Spectrometer)

Added requirements for:

- Spectral Flux Conversion

Additional Note:
SCAL/Telescope Correction requirements referred to the model-based implementation of this module. New requirements are required for the empirical based version of this processing module.

Changes in 1.1 (DLC):

Updated module assignments, including missing modules now assigned.

Updated Remove Correlated Noise requirements as received from K Xu

Updated Point Source Extraction requirements as received from Huw.

Updated Extract Level 0 Products, Add Nod and Raster Positions, Mask Bad Channels as received from Pasquale.

Added Time Correlation Correction (draft) requirements as received from Steve

Updated PCAL requirements as received from Dave Shupe

Updated Quality Control requirements as received from George.

Changes in 1.2 (DLC)

Updated Convert ADU to JFET voltage as received from Pasquale (14/7/08)

Updated Calculate Bolometer Voltage and Resistance from Pasquale (17/7/08)

Added Time Conversion and Reordering from Pasquale (18/7/08)

Added Add Nod and Raster Metadata updates from Pasquale (23/7/08)

Align with PDD – module titles (25/7/08)

Restructure to make bolometer, phtoometer and spectrometer chains clearer (25/7/08)

# Glossary

# 1. INTRODUCTION

The Herschel ICCs are responsible for providing software for the processing of observational data into standard products for distribution to observers. This software is run as a pipeline within the ESA SPG pipeline environment and consists of a series of modules called in sequence to progressively process the data towards a more scientifically useful product. AD01 describes the SPIRE pipline(s) in detail. This document takes the description of the pipeline steps in AD01 and extracts from this and from other applicable documents the detailed requirements for each of the SPIRE data processing pipeline modules

## 1.1 Scope

This document is intended to provide the full set of requirements for each SPIRE Data Processing Pipeline module. The details of the coding of each module to meet these requirements, their testing against these requirements and documentation on how to use these modules can be found elsewhere.

## 1.2 Structure

Section 2 lists all the modules in the SPIRE data processing pipeline as derived from AD01. Section 3 lists the global requirements for every pipeline module derived from both AD01 and AD02. Subsequent sections then describe the requirements for each individual module.

## 1.3 Documents

### 1.3.1 Applicable Documents

| | |
|---|---|
| AD01 | SPIRE Pipeline Description (SPIRE-RAL-DOC-002437) |
| AD02 | SPG Pipeline ICD |
| AD02 | Masks Policy Document |
| AD03 | SPIRE Bolometer Array Noise Performance (SVR-3.DOC.15) |
| AD04 | Darren Dowell's Note on Linearizaion Step and Temperature Drift Correction (Nov. 30, 2007) |
| AD05 | SPIRE Analogue Signal Chain and Photometer Detector Data Processing Pipeline (SPIRE-UCF-DOC-002890, Issue 4, 6 January 2008) |
| AD06 | SPIRE Spectrometer Pipeline Description (SPIRE-BSS-DOC-002966) |

### 1.3.2 Reference Documents

| | |
|---|---|
| RD01 | SPIRE Data Products Specification (SPIRE-RAL-DOC-002005) |

## 2. LIST OF MODULES

The list of processing modules in the SPIRE data processing pipeline is provided in Table 1, along with some basic information on the module's developer, which package it resides in and whether the module applies to photometer, spectrometer or both. A small number of modules have yet to be assigned to any individual developer and these are designated as TBD, while those assigned only preliminarily are designated TBC.

| | Module | Pipeline | Inst. | Responsible |
|---|---|---|---|---|
| 1 | Produce Data Frames from Packets | All | P,S | Pasquale |
| 2 | Extract Level 0 Products from data frames | All | P,S | Pasquale |
| 3 | Check ADC flags and Truncation | All | P,S | Pasquale |
| 4 | Mask Bad Channels | All | P,S | Pasquale |
| 5 | Convert ADU to JFET Voltage | All | P,S | Pasquale |
| 6 | Convert Non-Detector ADU to Engineering Values | All | P,S | Pasquale |
| 6a | Time correlation Correction | All | P,S | Huw |
| 7 | Time Conversion and Reordering | All | P,S | Pasquale |
| 8 | Calculate Bolometer Voltage and Resistance | All | P,S | Pasquale |
| 9 | First Level Deglitching | All | P,S | Christophe |
| 10 | Remove Electrical Crosstalk | All | P,S | CEA- Rene |
| 11 | Remove Optical Crosstalk | All | P,S | CEA - Rene |
| 12 | Extract Chop and Jiggle Positions | Phot. Jiggle | P,S | Rene |
| 13 | Calculate Pointing Timelines | All | P,S | Nieves |
| 14 | Associate Sky Position | All | P,S | Nieves (TBC) |
| 15 | Add Nod and Raster Positions | Phot. Jiggle | P,S | Pasquale |
| 16 | Remove Correlated Noise due to Bolometer Temperature Fluctuations | Phot. Scan | P | JPL |
| 17 | Subtract Operating Point Voltage | Phot. Scan | P | Not needed? |
| 18 | Correct Bolometer Time Response | All? | P,S | Cardiff - TBD |
| 18 a | Correct Electrical filter response | All? | P,S | Cardiff - TBD |
| 19 | Nonlinearity Correction and Correct Flux Density Conversion | Phot. Scan and Phot. Jiggle (different) | P | JPL |
| 20 | Demodulation | Phot. Jiggle | P,(S) | Rene |
| 21 | 2nd Level Deglitching and Averaging | Phot. Jiggle | P,(S) | Rene |
| 22 | Average Nod Cucles | Demodulation | P,(S) | Rene |
| 23 | De-Nod | Phot. Jiggle | P,(S) | Davide |
| 24 | Mapmaking | Map Making | P | Pierre |
| 25 | Clipping Correction | Spectrometer Proc | S | Dominique/LAM |
| 26 | Time Domain Phase Correction | Spectrometer Proc | S | Trevor |
| 27 | Interferogram Creation | Spectrometer Proc | S | Trevor |
| 28 | SCAL and Telescope Correction | Spectrometer Proc | S | Dominique/LAM |
| 29 | Interferogram Baseline Correction | Spectrometer Proc | S | Trevor |
| 31 | 2nd Level Deglitching (Spectrometer) | Spectrometer Proc | S | Trevor |
| 33 | Apodisation | Spectrometer Proc | S | Trevor |
| 34 | Fourier Transform | Spectrometer Proc | S | Trevor |
| 35 | Phase Correction | Spectrometer Proc | S | Trevor |
| 36 | Spectral Averaging | Spectrometer Proc | S | Trevor |
| 37 | Spectral Response Correction (Spectrometer) | Spectrometer Proc | S | Trevor |
| 38 | Spatial Regridding | Spectrometer Proc | S | Ralph |
| 40 | Quality Control | Quality Control | P,S | George |
| 41 | PCAL | PCAL | P,S | Arnie |
| 42 | Derive Point Source Flux Density and Position | Point Source | P | Davide |
| 43 | Point Source Extraction * | Point Source Extraction | P | Huw |

| | | | | |
|---|---|---|---|---|
| 44 | Visualize SPIRE detector timelines | Visualisation | P,S | Gabriele |
| 45 | Conversion to a Different Spectral Index | TBD (level 3) | P | TBD |

Table 1: List of SPIRE data processing Pipeline Modules

# 3. GLOBAL REQUIREMENTS

The following requirements apply to all modules. They are taken from AD02, AD02, ICC User Cases and ICC discussions.

## 3.1 General Requirements

**SPIRE -GEN-010** The module shall be written following the SPIRE pipeline module guidelines and HCSS developer guidelines
*The first document is TBD*
*This will provide e.g. information on how to issue error messages, how to use the command line to accept processing options and define input/output products*

| | | |
|---|---|---|
| **SPIRE -GEN-020** | The module shall be capable of being run within the SPG pipeline environment | |
| **SPIRE -GEN-030** | The module shall be capable of being run stand-alone in the Interactive Analysis environment<br>*I.e it should be possible to run it from the JIDE command line* | |
| **SPIRE -GEN-040** | The module shall allow all inputs and processing options to be specified on the command line<br>*A general specification for a common interface is given in TBD* | |
| **SPIRE -GEN-050** | The module shall include a GUI to allow a user to operate the module and select input products and parameters and output products without using the command line.<br>*It is expected that GIUs developed for the Interactive analysis shall conform to a common 'look and feel' to be determined by discussion within the HCSDT* | |
| **SPIRE -GEN-060** | The module shall be provided as a single DP task.<br>*The module may require more than one task to provide its functionality, but these should be 'hidden' within a single task 'wrapper'.* | |
| **SPIRE -GEN-070** | The module shall record the processing history in all output products<br>*DP tasks provide a mechanism for recording history. This shall include software versions, calibration file versions as well as the details of the processing parameters* | |
| **SPIRE -GEN-080** | On encountering an error that prevents further processing the module shall provide a meaningful error message and terminate gracefully<br>*This includes reporting invalid inputs* | |
| **SPIRE-GEN-090** | The module will check that the units of the input product are what is expected | |
| **SPIRE-GEN-100** | To use mask information to control processing in accordance with the Masks Policy document (AD03) | |

## 3.2 Quality Requirements

| | | |
|---|---|---|
| **SPIRE-QA-010** | The module shall provide quality metrics to allow the Quality Control Pipeline to assess the reliability of the output | |
| **SPIRE-QA-020** | The Quality Control metrics shall be stored in the output product metadata<br>*A set of acceptable values for these metrics will have to be provided for use by the Quality Control Pipeline* | |
| **SPIRE-QA-030** | The module shall provide quality information to allow the User to assess the reliability/quality of the output | |

## 3.3 Performance Requirements

**SPIRE-PERF-010** The module shall be able to process normal observations on a standard desktop machine
*The assumption is that the machine will have a minimum of 1 GByte of memory, 10 Gbyte of free hard disk space and a processor equivalent to a 1GHz Intel Core processor, TBD*

| | | |
|---|---|---|
| **SPIRE-PERF-020** | The module shall be able to process typical data on a standard desktop machine in less than 30 mins (TBC)<br>*The definition of typical data will vary from one AOT to another. However, all such typical observations should take less than 8 (TBC) hours for their observations* | |

## 3.4 Documentation Requirements

**SPIRE-DOC-010** The design of the module shall be documented in a Module Design Document

**SPIRE-DOC-020** There shall be a Programmers' Guide for the module
*This shall include a full description of the interfaces and implementation details*

**SPIRE-DOC-030** There shall be a User Guide for the module
*This shall include both standalone(GUI and command line) and pipeline use*

**SPIRE-DOC-040** There shall be a Test Plan for the validation and verification of the module.
*This shall detail the tests that will be performed to ensure that the requirements are fulfilled at the unit and system level*

**SPIRE-DOC-050** The module shall be delivered with a unit-level Test Report, which will show that the Validation and Verification tests have been passed

**SPIRE-DOC-060** Documentation of the code shall be through comments in the code and Javadoc statements conforming to the HCSS documentation standards

**SPIRE-DOC-070** Online documentation, including the User Guide, shall be provided in a form compatible with the HCSS Documentation Framework .
*This means using DocBook*

**SPIRE-DOC-080** Each delivery of a module shall be accompanied by a delivery note containing:

- Version number and release date
- List of files included in the delivery
- List of files changed for this release and their author(s)
- A summary of changes to the code
- List of SPRs and SCRs closed by this release

*Most of this information can be generated automatically from CVS Tags*

## 3.5 Testing Requirements

**SPIRE-TEST-010** The module shall be supplied with unit-level test harnesses
*These should test all interfaces and internal logic routing*

**SPIRE-TEST-020** The module shall be provided with the necessary input data to allow system tests as defined in the Test Plan to be carried out.
*This data does not have to be produced by the developer if it is available from elsewhere (e.g. other tests), however it shall be provided as part of the module delivery*

# 4. BOLOMETER PROCESSING CHAIN

## 4.1 Produce data frames from packets

### 4.1.1 Module Owner

CEA Saclay (Pasquale Panuzzo responsible).

### 4.1.2 Others Contributing

Steve Guest (RAL).

### 4.1.3 Module description

During SPIRE operations, instrument data are downloaded from the satellite (or from the test facility) in the form of telemetry (TM) packets and stored in a database. The data contained in each TM packet are stored as a bit array; thus in order to look at the values of telemetry data, the data must be reconstructed from the bit array. Using this design, it is not possible to query the database where the TM packets are stored for packets in which a given telemetry field has a selected value.

The implementation of SPIRE data frames is intended to provide a structure for TM data that is easily accessible (solving the above access limitation), and to allow easy conversion of data frames into SPIRE Data Products for data processing.

The On-Board Software (OBS) produces a number of different TM packet types. Some TM packet types are related to software/transmission errors or DPU events reports. These TM packets will be not translated into data frames because they are not directly related to scientific data. TM packet types that are related to detectors, BSM, SCU, SMEC, calibration sources and housekeeping have corresponding data frame types, one for each TM packet

type. In SPIRE telemetry, in order to reduce the number of TM packets, each packet can contain several science frames produced by the DRCU at different times. In the SPIRE definition of data frames, a SPIRE data frame instance corresponds to a single science frame from the DRCU. Moreover, each SPIRE data frame contains one member for each quantity contained in the parent TM packet. As an example, a science telemetry packet created by the SPIRE photometer could contain the voltage readouts of all 288 detectors taken at a number of times. This TM packet is converted into a number of data frames (one for each science frame contained in the packet) in which each voltage readout is stored in a specific member of the Java class 'data frame'.

The conversion of the TM packets to data frames will normally be done during data ingestion to the ICC database. However it shall be also possible to also create SPIRE data frames at processing time so that data processing can be done when data frames are not present in the database. The extraction of data frames from TM packets is done by the SPIRE Telemetry packet processor. The implementation of the SPIRE TM packet processor has to bear in mind performance issues. If the SPIRE TM packet processor is not efficient enough, there could be a loss of data during the ingestion of data frames to the ICC database. Thus the SPIRE Telemetry packet processor shall be able to process a TM packet in a few (<10) milliseconds.

The SPIRE TM packet processor shall also check the integrity and validity of TM packets and the frames inside them. The integrity of TM packets is checked using the cyclic redundancy check (CRC), rejecting those with an invalid value. The frames inside TM packets are normally checked by the OBS, so there is no real need to check them with the packet processor. However, the packet processor shall nevertheless analyze the checksum value and the frame ID value validity for all frames and will reject those data frames from an affected packet. Allowing frames through that fail the checks would cause errors in the OBS or in the SPIRE software.

This module is responsible for developing SPIRE data frame classes and the SPIRE Telemetry packet processor.

## 4.2 Extract Level 0 Products from data frames

### 4.2.1 Module Owner

CEA Saclay (Pasquale Panuzzo responsible).

### 4.2.2 Others Contributing

Currently none.

### 4.2.3 Module description

This module is responsible to extract telemetry data from the Versant database and to compile them into a set of Level 0 products.

The telemetry data are supposed to be in form of spire data frames or telemetry packets. In the latter case, this module will use the step described in the previous section to convert telemetry packets into data frames. Telemetry packets for which data frames are not defined will be ignored.

The user will specify which data shall be extracted by specifying a time range, an OBSID, or an OBSID and BBID. The connection with the database shall be done by using the Access package API or directly via an ObjectStore.

The telemetry data are stored and grouped into a number of level 0 products. Each level 0 product will contain only data coming from a single building block, i. e. they will have the same OBSID and BBID. Each level 0 product will contain only data of the same kind; thus a number of different products will be created from a single building block. The Spire product description document lists the expected products for each telemetry packet type.
Level 0 products have the following structure. They contain a single table dataset with one column for each telemetry parameter. The column is named with the QLA name of the parameter. The telemetry parameter values are stored in their raw form.

The order in which data are stored into raw data products is as the data frame extraction. So it is not guaranteed that they will be time ordered. Moreover, some data frame timestamps could be invalid and they will need to be checked and reprocessed in the following steps.

### 4.2.4 Data

#### *4.2.4.1 Input Data*

The input data will be in the form of telemetry packets or data frames.

#### *4.2.4.2 Input Calibration Products*

No calibration product is needed for this module.

#### *4.2.4.3 Output Data*

The output data will be in the form of level 0 products.

## 4.3 Reformat Level 0 Products into Level 0.5 Products

### 4.3.1 Module Owner

CEA Saclay (Pasquale Panuzzo responsible)

### 4.3.2 Others Contributing

Currently none.

### 4.3.3 Module description

This module is responsible for reformatting level 0 products into a more convenient form. The Level 0 Products contain a single TableDataset with one column for telemetry parameters. The name of each column is the QLA telemetry parameter name. The level 0.5 product shall have a table for signals and a table for mask values. The columns shall use more friendly names such as the real names of channels (e. g. “PSWA8”).

## 4.4 Time Correlation

This module applies a time correction from on-board-time (OBT) to TAI. To quote **[RD1]**, “The accuracy of the correlation affects science data processing…”

Reference Documents:
**[RD1]:** H/P OBT-UTC Time Synchronisation Technical Note, PT-CMOC-OPS-TN-6604-OPS-OGH, R.Furnell, Issue 2 December 2004.
**[RD2]**: Herschel-Planck Mission Control System – Interface Control Document – Time Correlator, PT-CMOC-MDS-ICD-3102-OPS-GDS, Tonny Ulriksen, Issue 2.4 24th January 2008.
**[RD3]**: Herschel Auxiliary Products Specification, HERSCHEL-HSC-DOC-0816, Miguel Sanchez-Portal, Issue 0.7 Draft, 25th January 2008.

### 4.4.1 Module Owner

TBD

### 4.4.2 Others Contributing

TBD

### 4.4.3 Module Description

This module applies a time correction from on-board-time (OBT) to TAI. These times are also known as *uncorrelated* and *correlated* times respectively. The conversion from TAI to UTC is well-known, well-defined, and out of the scope of this module.

### 4.4.4 Assumptions

- The input times to be corrected in the data products are in 64-bit double precision format, expressed as seconds since Jan 1st 00:00 1958. Nevertheless, the module shall also be capable of handling long integer format, see TIMECORR-FUN-030.
- The outputs of this module in the SPG are the Level-1 products.
- The time correlation product already contains the results of a least square fit of time pair data points. In other words, the module does not have to perform such a fit itself.

### 4.4.5 Input Data

- SPIRE data products with time in OBT.
- Time Correlation Product

### 4.4.6 Output Data

- SPIRE data products with times in TAI.

### 4.4.7 Functional Requirements

| | |
|---|---|
| **TIMECORR-FUN-010** | The module shall convert the times in the input products from OBT to TAI. Both the sample times in the columns and the time range in the metadata (startDate/endDate) shall be converted. |
| **TIMECORR-FUN-020** | The reverse transformation shall also be possible i.e. TAI to OBT. |
| **TIMECORR-FUN-030** | The module shall be able to convert times in both 64-bit long integer (microseconds since 1958) and double precision (seconds since 1958) formats. |
| **TIMECORR-FUN-040** | The module shall set a metadata item to indicate which time is contained in the product. (Is there a standard one for this?). |

### 4.4.8 Non-Functional Requirements

| | |
|---|---|
| **TIMECORR-NON-010** | The maximum permitted absolute error in the corrected time is **TBD**. Note that the Herschel Ground Segment requirement is 20ms. |
| **TIMECORR-NON-020** | Performance requirement is **TBD** but significant. The photometer detectors alone produce over 50,000 time samples per hour, all of which have to be corrected. |

## 4.5 Check ADC flags and Truncation

### 4.5.1 Module Owner

CEA Saclay (Pasquale Panuzzo responsible).

### 4.5.2 Others Contributing

Currently none.

### 4.5.3 Module Description

This module shall check if the ADC flag values are zero or not. In the case that the value is not zero, the module shall flag the sampling of the affected channels by setting the appropriate bit of the mask. The fraction of flagged samples shall be recorded. If any flag is different from zero in the detector timeline, this shall be recorded in a quality control metadata keyword.

This module shall also check if detector channel values are truncated, i.e. if the voltage is out of the ADC range. When a detector channel voltage is out of the ADC range, the measured ADU value is 0 or 65535; when these values are found, the sampling shall be flagged as invalid by setting the appropriate bit of the mask. The fraction of flagged sampling shall be recorded.

### 4.5.4 Data

#### *4.5.4.1 Input Data*

The input data will be in the form of a detector timeline.

#### *4.5.4.2 Input Calibration Products*

No calibration product is needed for this module.

#### *4.5.4.3 Output Data*

The output data will be in the form of a detector timeline.

## 4.6 Mask Bad Channels

### 4.6.1 Module Owner

CEA Saclay (Pasquale Panuzzo responsible).

### 4.6.2 Others contributing

Currently none.

### 4.6.3 Module Description

This module shall
1) properly flag in detector timelines all samples of channels that are identified as dead or noisy channels in a ChanMask calibration product.
2) remove disconnected channels from detector timelines which are identified as not connected in a ChanNum calibration product.

### 4.6.4 Data

#### *4.6.4.1 Input Data*

The input data will be in the form of a detector timeline.

#### *4.6.4.2 Input Calibration Products*

The channel names that are dead or noisy are identified in the SCalPhotChanMask & ScalSpecChanMask calibration products. The disconnected channels are identified in the SCalPhotChanNum & ScalSpecChanNum calibration products.

#### *4.6.4.3 Output Data*

The input data will be in the form of a detector timeline.

## 4.7 Convert ADU to JFET Voltage

### 4.7.1 Module Owner

CEA Saclay (Pasquale Panuzzo responsible).

### 4.7.2 Others Contributing

Currently none.

### 4.7.3 Module Description

This module takes the ADU values of detectors readouts, combines them with the offset history file and the electronic gain table, and uses them to calculate the voltages across each channel. The voltages resulting from this module are the RMS voltages at the JFET output. The processing in this module is common to all SPIRE pipelines.

### 4.7.4 Data

#### *4.7.4.1 Input Data*

The input data shall be in the form of a detector timeline. The module also needs the bias frequency that was used during the measurement. The bias frequency to be used shall be passed to the module by the user as 1) a parameter, 2) providing a nominal housekeeping timeline, or 3) setting the value of the bias frequency in the metadata of the input detector timeline.

#### *4.7.4.2 Input Calibration Products*

The module requires channel gain calibration products (SCalPhotChanGain & SCalSpecChanGain) and the signal offset history calibration product. The latter will have been produced by the Operational Day Processing.

#### *4.7.4.3 Output Data*

The output data produced by this module shall be in the form of a detector timeline. The module shall set the value of the used bias frequency in the metadata of the output product. The module shall also record in the metadata if the offset was used.

### 4.7.5 Functional Requirements

| | |
|---|---|
| **JFETCON-FUN-010** | The module shall operate on data in the form of a detector timeline. |
| **JFETCON-FUN-020** | The module shall output data in the form of a detector timeline. |
| **JFETCON-FUN-030** | The module will use the Offset History File to determine the value of the signal offset. |
| **JFETCON-FUN-040** | The module shall use an SCalPhotChanGain calibration product for photometer modes and an SCalSpecChanGain calibration product for spectrometer modes to get total gain of DCU chain at the specified bias frequency |
| **JFETCON-FUN-050** | When the Offset History is provided, the module shall calculate the channel voltage using the equation:<br><br>$V_{JFET-RMS}=[5/G_{tot}(\omega_b)]*[(ADU-2^{14}+52428.8*OFF)/(2^{16}-1)]$ (Volts)<br><br>where $\omega_b=2*\pi*f_b$ where $f_b$ is the bias frequency in Hertz.<br>This formula is Equation 17 of AD03. |
| **JFETCON-FUN-060** | When the Offset History is not provided, the module shall calculate the channel voltage using the equation:<br><br>$V_{JFET-RMS}=[5/G_{tot}(\omega_b)]*[ADU/(2^{16}-1)]$ (Volts)<br><br>where $\omega_b=2*\pi*f_b$ where $f_b$ is the bias frequency in Hertz. |
| **JFETCON-FUN-070** | The voltages in the output product shall be in float (single precision). |

# 5. Convert Non-Detector ADU to Engineering Values

### 5.1.1 Module Owner

CEA Saclay (Pasquale Panuzzo responsible).

### 5.1.2 Others Contributing

Currently none.

### 5.1.3 Module Description

This module is responsible for converting raw ADU values into engineering values for timelines other than detector timelines, e.g. housekeeping timelines, SCU timelines etc. The conversions are done with QLA conversion tables.

## 5.2 Time Conversion and reordering

### 5.2.1 Module Owner

CEA Saclay (Pasquale Panuzzo responsible).

### 5.2.2 Others Contributing

Currently none.

### 5.2.3 Module Description

This module is responsible for converting the time in timelines and for the reordering of timelines. This module is common to all pipelines and all timelines shall be processed with it.

### 5.2.4 Data

#### *5.2.4.1 Input Data*

The input data shall be in the form of a timeline product.

#### *5.2.4.2 Input Calibration Products*

The module requires the DPU reset history product (SCalResetHist) to process science data timelines (detector timelines, offset timelines, BSM timelines, SCU timeline, SMEC timelines and Test Fourier Transform Spectrometer Science timelines). The module doesn't require any calibration product for Nominal Housekeeping, Critical Housekeeping, Test Facility Control System Housekeeping, and Test Fourier Transform Spectrometer Housekeeping timelines.

#### *5.2.4.3 Output Data*

The output product shall have the same format of input product, with the sample time converted in seconds from 1958 epoch. The resulting sample time is intended to be in the on board time reference, i.e. not corrected for time drift of spacecraft clock respect to the universal time.

### 5.2.5 Functional Requirements

| | |
|---|---|
| **TIMECONV-FUN-010** | The module shall operate on science and HK timelines. |
| **TIMECONV-FUN-020** | The module shall use the DPU Reset History (SCalResetHist) to determine the DPU reset time for all science data timelines. |
| **TIMECONV-FUN-030** | For Housekeeping timelines, the module shall compute the sample time from the packet time using the equation:<br>sampleTime=packetTime/65536.0 |
| **TIMECONV-FUN-040** | For Housekeeping timelines, the module shall sort the “signal” and “mask” tables using the sampleTime. |
| **TIMECONV-FUN-050** | For science data timelines, for each frame, the module should get the appropriate reset time from the reset history using the packetTime, then use it to compute the sampleTime as:<br><br>$sampleTime=reset/65536.0+frameTime*clockTick+2^{32}*rolls*clockTick$<br><br>where clockTick is the clock tick in seconds (value taken from param tables) and rolls is the number of estimated rolls over of the frameTime counter. |
| **TIMECONV-FUN-060** | The module shall estimate the number of rolls of the frameTime counter as the number for which sampleTime < packetTime and sampleTime > packetTime-10min. If no value for roll is found, the module shall mark the frame as having an invalid time and assume roll=0. |
| **TIMECONV-FUN-070** | The output timeline shall have the sampleTime time-ordered. |
| **TIMECONV-FUN-080** | The module shall remove the columns “frameTime”, “sdfTime”, “packeTime” and “seqCount” from the output product. |

## 5.3 Calculate Bolometer Voltage and Resistance

### 5.3.1 Module Owner

CEA Saclay (Pasquale Panuzzo responsible).

### 5.3.2 Others Contributing

René Gastaud (CEA Saclay)

### 5.3.3 Module Description

This module is responsible for computing the RMS detector voltages and resistances starting from the JFET output voltage. The processing in this module is common to all SPIRE pipelines.
The detector voltages and resistances shall be computed using the iterative algorithm described in AD05.

### 5.3.4 Data

#### *5.3.4.1 Input Data*

The input data shall be in the form of a detector timeline. The voltages in the input detector timeline are assumed to be the RMS voltage at the JFET output. The module also needs the bias frequency and the bias voltage amplitudes for the computation. The bias amplitudes shall be passed to the module by providing a Nominal Housekeeping Timeline. The bias frequency shall be obtained from the metadata of input detector timeline, or alternatively from the Nominal Housekeeping Timeline.

#### *5.3.4.2 Input Calibration Products*

The module requires channel gain calibration products (SCalPhotChanGain & SCalSpecChanGain) and the Bolometer Parameter table (SCalPhotBolPar & SCalSpecBolPar). The channel gain calibration products are needed to obtain the JFET gain. The Bolometer Parameter table is needed to get the load resistances and the harness cable capacities. The module shall use, if provided, the Channel Nominal Resistances table (SCalPhotChanNomRes & SCalSpecChanNomRes) to get the nominal resistance of each channel.
The module shall also use the Channel Number Mapping calibration product (SCalPhotChanNum & SCalSpecChanNum), if provided, to obtain which channels are Temperature Control, whose voltage shall not be processed by this module.

#### *5.3.4.3 Output Data*

The output data produced by this module will be in the form of a detector timeline. The output detector timeline shall contain a table with voltages, a table with resistances and a table with phase shifts. The output timeline shall also contain in metadata the bias amplitudes.

### 5.3.5 Functional Requirements

| | |
|---|---|
| **VOLTRES-FUN-010** | The module shall operate on data in the form of a detector timeline. |
| **VOLTRES-FUN-020** | The module shall output data in the form of a detector timeline. |
| **VOLTRES-FUN-030** | The module shall use an SCalPhotChanGain calibration product for photometer modes and an SCalSpecChanGain calibration product for spectrometer modes to get the JFET gain. |
| **VOLTRES-FUN-040** | The module shall use an SCalPhotBolPar calibration product for photometer modes and an SCalSpecBolPar calibration product for spectrometer modes to get the load resistance, the capacity of the harness |
| **VOLTRES-FUN-050** | The module shall use, when provided, an SCalPhotChanNomRes calibration product for photometer modes and an SCalSpecChanNomRes calibration product for spectrometer |

modes to get the nominal resistance of the channels

**VOLTRES-FUN-060** The module shall use, when provided, an SCalPhotChanNum calibration product for photometer modes and an SCalSpecChanNum calibration product for spectrometer modes to know which are the Thermal Control channels

**VOLTRES-FUN-070** When the Channel Nominal Resistances product is provided, the module shall calculate the channel voltage, the resistance and the phase shift using the equations:

$V_{d\text{-}rms}=V_{JFET\text{-}RMS}/[H_{JFET}*|H_H(\omega_b)|*\cos(\Delta\varphi)]$ (Volts)

$R_d=(V_{b\text{-}rms}/I_{b\text{-}rms})-R_L$ (Ohm)

$\Delta\varphi=\tan^{-1}(\omega_b*\tau_{H\text{-}nom})-\tan^{-1}(\omega_b*\tau_H)$

where $\omega_b=2*\pi*f_b$ where $f_b$ is the bias frequency in Hertz, and

$|H_H(\omega_b)|=1/[1+(\omega_b*\tau_H)^2]^{1/2}$

$\tau_H=C_H*[R_L*R_d/(R_L+R_d)]$,

$\tau_{H\text{-}nom}=C_H*[R_L*R_{d\text{-}nom}/(R_L+R_{d\text{-}nom})]$,

$I_{b\text{-}rms}=(V_{b\text{-}rms}-V_{d\text{-}rms})/R_L$

**VOLTRES-FUN-080** The channel voltage, the resistance and the phase shift shall be computed with an iterative procedure. The final value shall be within 0.1% from the analytical value.

**VOLTRES-FUN-090** When the Channel Nominal Resistances product is not provided, the module shall calculate the channel voltage and the resistance using the equations:

$V_{d\text{-}rms}=V_{JFET\text{-}RMS}/[H_{JFET}*|H_H(\omega_b)|]$ (Volts)

$R_d=(V_{b\text{-}rms}/I_{b\text{-}rms})-R_L$ (Ohm)

where $\omega_b=2*\pi*f_b$ where $f_b$ is the bias frequency in Hertz, and

$|H_H(\omega_b)|=1/[1+(\omega_b*\tau_H)^2]^{1/2}$

$\tau_H=C_H*[R_L*R_d/(R_L+R_d)]$,

$I_{b\text{-}rms}=(V_{b\text{-}rms}-V_{d\text{-}rms})/R_L$

**VOLTRES-FUN-100** The voltages of TC channels in the output product shall be equal to the input one. Resistances and phase shits for TC channels in the output product shall be set to NaN.

**VOLTRES-FUN-110** The voltages, resistances and phase shifts in the output product shall be in float (single precision).

## 5.4 First Level (Time Domain) Deglitching

## 5.5 14.1 Module Owner

LAM (Christophe Ordénovic)

## 5.6 14.2 Other contributions

BlueSkySpectroscopy (Trevor Fulton, Peter-Davis Imhof)

## 5.7 14.3 Module Description

This module identifies and removes glitches from the spectrometer or photometer detector timelines or from the spectrometer detector interferograms. The method employed merges a local regularity analysis with a wavelet analysis. A general description of the method can be found in C.Ordenovic, C. Surace, B.Torresani, A. Llebaria, JP. Baluteau, Use of a local regularity analysis by a wavelet analysis for glitches detection, Proc. SPIE Vol. 5909, pp556-567, 2005.

### 5.7.1 14.4 Data

#### *5.7.1.1 14.4.1 Input data*

The input data can be in the form of a spectrometer or photometer detector timeline or a spectrometer detector interferogram. The data in the detector timeline will have been processed so that bad channels and parts of the data where ADC flags indicate out of range or otherwise corrupted data have been masked.

The SpireMask product will be used to keep trace of which samples are flagged as glitches and which glitches have been repaired.

#### *5.7.1.2 14.4.2 Input Control Parameters*

These numerical parameters are optional and have been set in order to give the best performance for glitch detection for PDT, SDT or spectrometer interferograms. They should not be modified from the values optimised for each of these products.

- scaleMin and ScaleMax define the scale range used by the algorithm to perform the Holder estimation by using a linear regression. These values are respectively fixed to 2 and 8 for SDTs
- thresoldHolder and hMin defines the range where the local Holder exponent is flagged as a glitch. These values are respectively fixed to -0.6 and -1.4 for SDTs
- thresholdCorr defines the lower limit on the correlation coefficient to perform the Holder estimation. This value is fixed to 0.975 for SDTs

A flag parameter (reconstruction) if true causes the module to perform signal reconstruction by removing the detected glitches. The value is set at 'true' by default.

#### *5.7.1.3 14.4.3 Output Data*

The output data produced by this module will be in the same form as the input data (spectrometer detector timeline or spectrometer detector interferogram).

### 5.7.2 14.5 Functional requirements

| | | |
|---|---|---|
| **DEG-FUN-010** | The module shall operate on data in the form of a spectrometer or photometer detector timeline or a spectrometer detector interferogram | |
| **DEG-FUN-020** | The module shall output data in the same form as the input data | |
| **DEG-FUN-030** | The module shall use the SpireMask product to register the flagged samples | |
| **DEG-FUN-040** | On each detector timeline :<br>-the module shall perform a wavelet decomposition<br>-the module shall extract the extrema representation (Wavelet Trasform Modulus Maxima Line, WTMML). For small scales, each maxima line converges toward a sample of the signal. | |

| | | |
|---|---|---|
| | -for each of these samples :<br>-on a (log W-log s) representation (W and s respectively being the wavelet coefficient and the scale of decomposition), the module shall perform a linear regression over the range (scaleMin, scaleMax) and will estimate its coeffieicnt correlation C : if c > thresholdCoeff, the Holder exponent h is given by the slope of the regression.<br>Finally, if h is in the range (thresHolder, hMin), the corresponding sample is flagged as a glitch. | |
| **DEG-FUN-050** | flagged samples shall be stored into the SpireMask product | |
| **DEG-FUN-060** | If the 'reconstruction' parameter is set to 'true', the module shall perform a glitch removal over all flagged samples. It will do this by locally reconstructing the signal with the glitch wavelet components removed. | |

## 5.8 Remove Electrical Crosstalk

## 5.9 Remove Optical Crosstalk

## 5.10 Extract Chop and Jiggle Positions

This module is one of the pre-processing steps used to process Beam Steering Mirror information. It shall identify the different positions of a jiggle map in the chopper sensor timeline and jiggle sensor timeline. This will help to demodulate each bolometer, and then to compute a jiggle map.

### 5.10.1 Module Owner

CEA Saclay (René Gastaud responsible).

### 5.10.2 Others Contributing

Currently none.

### 5.10.3 Module Description

The purpose of this module is to process Beam Steering Mirror angles timelines and extract the positions of a jiggle map: for each position we compute a start time and an end time. This will be used by subsequent pipeline processing steps. The position Y angle and Z angle in degrees of each jiggle map position will also be computed for completeness. This is a not a duplication of BSMConverter task, because the module calculates the position of each node in the BSM operation table, and not the actual position.

### 5.10.4 Data

#### *5.10.4.1 Input data*

There is one input data product, which includes Beam Steering Mirror Chopping Sensor and Jiggle Sensor timelines. This is described in the SPIRE Data Products Specification as Beam Steering Mirror Timeline (BSMT).

#### *5.10.4.2 Input Calibration Product*

There are two calibration products.

- BSM Operations Table: this is described in SPIRE Data Products Specification as DPCR-09. It will include the Beam Steering Mirror Chopping Sensor and Jiggle Sensor values for each position of each jiggle map, with the range of allowed variation. These values are in ADU. This is used to identify the positions in the jiggle map and for the chopper On/Off positions.
- BSM Position Table: this is described in SPIRE Data Products Specification as DPCR-10. It will include the Beam Steering Mirror Chopping Sensor and Jiggle Sensor values in ADU and their counterpart Y Angle and Z Angle in degrees. It will also include the authorized range of angles. It is used for the conversion from adu to decimal degree.

#### *5.10.4.3 Output data*

There is only one output data product. It is described in the SPIRE Data Products Specification as Chop and Jiggle Timeline (CJT). It will contain a column Chopper Identity, a column Jiggle Identity, a column Start Time, a column End Time. It will also contain a column Chopper Time which gives the time of the motion of the chopper, and a column Chopper Flags which gives the sign of the motion (going left or going right).

#### *5.10.4.4 Functional Requirements*

| | |
|---|---|
| **BSMFlagsExtractor-FUN-010** | The module shall process data from a single BSMT |
| **BSMFlagsExtractor-FUN-020** | The module shall find the start time and end time where the Beam Steering Mirror is in a defined jiggle map position |
| **BSMFlagsExtractor-FUN-030** | The calculated quantities are:<br>start time and end time for each defined jiggle map position. |
| **BSMFlagsExtractor-FUN-040** | The module shall report missing positions of the jiggle map. |
| **BSMFlagsExtractor-FUN-050** | The module shall find the time of each motion of the chopper. |
| **BSMFlagsExtractor-FUN-060** | The calculated quantities are:<br>Chopper Time and Chopper Flag |
| **BSMFlagsExtractor-FUN-070** | The module shall convert the angles using the best method (linear interpolation, spline interpolation). |
| **BSMFlagsExtractor-FUN-080** | The calculated quantities are:<br>Y Angle and Z Angle in decimal degree for each node of the chopper/ jiggle map. |

## 5.11 BSM angles conversion

This module is one of the processing steps used to process data from the detectors. It shall compute the angles ..... This will help to compute the actual pointing of each bolometer during an observation.

### 5.11.1 Module Owner

CEA Saclay (René Gastaud responsible).

### 5.11.2 Others Contributing

Currently none.

### 5.11.3 Module Description

The purpose of this module is to convert the angles of the Beam Steering Mirror into physical units. This will be used by subsequent pipeline processing steps.

### 5.11.4 Data

#### *5.11.4.1 Input data*

There is only one input data product, which will include Beam Steering Mirror Chopping Sensor and Jiggle Sensor timelines. This is described in SPIRE Data Products Specification as Beam Steering Mirror Timeline (BSMT).

#### *5.11.4.2 Input Calibration Product*

There is only one calibration product. It will include Beam Steering Mirror Chopping Sensor and Jiggle Sensor values in ADU and their counterpart Y Angle and Z Angle in degrees. It will also include the authorized range of angles. This is described in SPIRE Data Products Specification as DPCR-10 BSM Position Table.

#### *5.11.4.3 Output data*

There is only one output data product, which will contain Beam Steering Mirror Chopping Sensor and Jiggle Sensor timelines in decimal degree on the sky. This is described in SPIRE Data Products Specification as Beam Steering Mirror Angles Timeline (BAT).

#### *5.11.4.4 Functional Requirements*

| | |
|---|---|
| **BSMConverter-FUN-010** | The module shall process data from a single BSMT |
| **BSMConverter-FUN-020** | The module shall convert the angles using the best method (linear interpolation, spline interpolation). |
| **BSMConverter-FUN-030** | The calculated quantities are:<br>Y Angle and Z Angle in decimal degrees. It will be stored in a BAT. |
| **BSMConverter-FUN-040** | The module shall report angles out of the nominal range. |

## 5.12 Calculate Pointing Information

### 5.12.1 Module Owner

Instituto de Astrofísica de Canarias, IAC (Nieves Castro-Rodríguez responsible)

### 5.12.2 Others Contributing

Pasquale Panuzzo (CEA Saclay).

### 5.12.3 Module Description

The purpose of this module is to provide information on pointing for SPIRE data using the information from the BSM position, the global spacecraft pointing, the relative position of each pixel, and the information of the SIAM matrix (Spacecraft-Instrument Alignment Matrix). This module is common to all SPIRE pipelines and works for both the photometer and spectrometer.

## 5.12.4 Data

### *5.12.4.1 Input Data*

The input data will be products.

The SpirePointingProduct (this module) has a Map Context structure. We have created a collection where every object contained in the map is identified with a *string*. With this structure, we encapsulate the spacecraft pointing product and move it to the Spire Pointing Product in an easy way.

This module uses two data products as input data:

- a BsmAngleTimeline (which contains the position of the BSM as a function of time)
- a Herschel PointingProduct (the spacecraft pointing)

### *5.12.4.2 Input Calibration Products*

The input calibration products are:

- a PixAngOff product (the calibration product that contains the position of each pixel with respect to the centre of the array)
- a SiamProduct (contains the information about the orthogonal rotation matrix which converts the coordinates of a vector in the spacecraft reference frame into a given instrument frame, in this case for SPIRE).

These calibration products are derived from other pipeline modules.

### *5.12.4.3 Output Data*

The output data produced by this module will be in the form of a SpirePointingProduct.

## 5.12.5 Functional Requirements

**SPOINT-FUN-010** The module shall operate on products.

**SPOINT-FUN-020** The module shall operate in a Map Context. We use the Context class that is *a product of products* to encapsulate the spacecraft pointing product.

**SPOINT-FUN-030** The module operates on the photometer and spectrometer data.

**SPOINT-FUN-040** The module should use a PointingProduct, a SiamProduct, a PinxAngOff and a BsmAngleTimeline

**SPOINT-FUN-050** The module output is a product (SpirePointingProduct).

**SPOINT-FUN-060** The module defines methods to check and set the following values:

- Calculate the BSM y-angle and z-angle for a given time or for a time array.
- Calculate the offsets of each pixel with respect to the centre of the array.
- Convert the SIAM product to a 3x3 matrix.
- Calculate the SampleTime using information in BsmAngleTimeline.

**SPOINT-FUN-070** The module shall calculate the position in the sky of each detector unit or pixel using a getSkyPosition method. This method uses the pixel offsets, the BSM angles and the SIAM matrix for the instrument, and calculates the relative position of the pixels with respect to the system (using the spacecraft pointing).

| | |
|---|---|
| **SPOINT-FUN-080** | The modules needing to know the position of a pixel at a given time will ask this product to compute the correct position using one of the above methods. |
| **SPOINT-FUN-090** | The module uses a filtered spacecraft pointing. *We have to check which is better: filtered o gyropropagated.* |
| **SPOINT-FUN-100** | Where the inputs are wrong the module gives an IOException. |

## 5.13 Associate Sky Position

### 5.13.1 Module Owner

Instituto de Astrofísica de Canarias, IAC (Nieves Castro-Rodríguez responsible)

### 5.13.2 Others Contributing

Pierre Chanial (Imperial College), Pasquale Panuzzo (CEA Saclay).

### 5.13.3 Module Description

The purpose of this module is to provide information on the equivalent position for each pixel as a function of time. This task gives a result using the pixel positions computed in the SpirePointingProduct module (see *Calculate Pointing Information* in this document).

### 5.13.4 Data

#### *5.13.4.1 Input Data*

The input data will be a table dataset and a product:

- DetectorTimeline (dataset with information about the detector properties).
- SpirePointingproduct (uses the position of the BSM, the spacecraft pointing, the SIAM matrix and the relative offsets of the pixels and gives a product with the pointing information for SPIRE.).

#### *5.13.4.2 Output Data*

The output data produced by this module will be a new DetectorTimeline with information about the position of each pixel at different times.

### 5.13.5 Functional Requirements

| | |
|---|---|
| **ASP-FUN-010** | The module shall operate on a product (SpirePointingProduct) and a dataset (DetectorTimeline). |
| **ASP-FUN-020** | The module shall operate as a Task. |
| **ASP-FUN-030** | The module operates on photometer and spectrometer data. |
| **ASP-FUN-040** | The module shall return a DetectorTimeline. |
| **ASP-FUN-050** | The module shall calculate the position in the sky for each pixel at a given sample time using a SpirePointingProduct. |
| **ASP-FUN-060** | The module gives an IOException if the inputs are wrong |

## 5.14 Add Pointing Metadata Parameters

CEA Saclay (Pasquale Panuzzo responsible)

### 5.14.1 Others Contributing

Currently none.

### 5.14.2 Module Description

This module is responsible for computing the pointing information and for adding this information in metadata of detector timelines. The pointing information are: the nodding ID for photometer Point Source and Small Map observations, the scan line number for Large Map and parallel observations, the jiggle ID and pointing number for spectrometer observations. The pointing information is stored the nominal housekeeping parameter STEP. The description of which bits in STEP are used to store the above information is specified in the SPIRE AOT Implementation Document (SPIRE-RAL-DOC-002663).

### 5.14.3 Data

#### *5.14.3.1 Input Data*

The input data shall be in the form of a detector timeline. The module needs the value of the STEP parameter. This value shall be provided to the module by passing directly the value or by providing a Nominal Housekeeping Timeline.

#### *5.14.3.2 Input Calibration Products*

The module doesn't require any calibration product.

#### *5.14.3.3 Output Data*

The output data produced by this module shall be in the form of a detector timeline with the pointing information stored in metadata parameters.

### 5.14.4 Functional Requirements

| | |
|---|---|
| **ADDPOINT-FUN-010** | The module shall operate on data in the form of a detector timeline. |
| **ADDPOINT-FUN-020** | The module shall output data in the form of a detector timeline. |
| **ADDPOINT-FUN-030** | The module shall compute pointing information from the STEP parameter. It shall be possible to pass the STEP value as a module parameter or alternatively providing a Nominal Housekeeping Timeline. |
| **ADDPOINT-FUN-040** | For instrument modes POF2 and POF3, the module shall compute the nodding ID, as the value of the bits number 14 of STEP. |
| **ADDPOINT-FUN-050** | For instrument modes POF5 and POF9, the module shall compute the scan line number, as the value of the bits from 0 to 13 of STEP. |
| **ADDPOINT-FUN-060** | For spectrometer instrument modes, the module shall compute the Jiggle ID, as the value of the bits from 0 to 6 of STEP. |
| **ADDPOINT-FUN-070** | For instrument modes SOF1R and SOF2R, the module shall compute the Point Number, as the value of the bits from 7 to 13 of STEP. |

## 5.15 Remove Correlated Noise due to Bolometer Temperature Fluctuations

## 5.16 REMOVE CORRELATED NOISE DUE TO BOLOMETER TEMPERATURE FLUCTUATIONS

This module is part of both the SPIRE Empirical Photometer Pipeline and SPIRE Spectrometer Pipeline. It processes detector time lines produced by SPIRE Photometer in the scan map mode (POF5) and by the SPIRE Spectrometer in continuous scan modes (SOF1 and SOF2). The module subtracts correlated low frequency (< 1Hz) noise caused by variations of the detector bath temperature. For scan map observations, the low frequency noise can be dominant over the white noise. Other observations that invoke short term, higher frequency modulations do not need this correction. This module is not in the Model-based Pipeline, where the dependence of

the detector voltage on the bath temperature has been taken into account in the module that converts the detector voltage to the optical power.

### 5.16.1 Module Owner

NHSC (Arnie Schwartz, implementation; C. Kevin Xu, science requirements).

### 5.16.2 List of Applicable Documents

| | |
|---|---|
| AD01 | SPIRE Pipeline Description (SPIRE-RAL-DOC-002437) |
| AD02 | SPG Pipeline ICD |
| AD03 | SPIRE Bolometer Array Noise Performance (SVR-3.DOC.15) |
| AD04 | Darren Dowell's Note on Linearizaion Step and Temperature Drift Correction (Nov. 30, 2007) |

### 5.16.3 Module Description

The purpose of this module is to subtract low frequency (< 1Hz) noise, caused by variations of the detector array bath temperature, from scan mapping data time lines. The module shall be run after the nonlinearity correction module which also converts the detector signals into Jy (maybe different for Spectrometer). The basic assumptions for the module algorithm are:
(1) The bolometer channels of the SPIRE Photometer are stable in terms of responsivity to the optical load.
(2) The responsivity of themisters and dark pixels to the detector bath temperature does not change with time.
(3) The bias voltage is at one of the two nominal values.
(4) The gamma function, $\gamma_i(V_T) = f_i \times dV_i/dV_T$ , can be specified by a linear function of the thermister voltage $V_T$: $\gamma_i(V_T) = A_i + B_i \times (V_T - V_{T0})$, where $V_{T0}$ is the nominal detector array thermister signal, and $f_i=dS/dV_i$ a measure of responsivity. The gamma function is independent of optical load, but changes with the bias voltage. The correction is then estimated by the following formula:
$\underline{Q_{T,i}} = A_i \times (\underline{V_T} - V_{T0}) + 0.5 \times B_i \times (\underline{V_T} - V_{T0})^2$.
(5) At the high bias voltage, the thermisters are saturated, and the temperature drift will be traced by dark pixels. Therefore, for the high bias voltage, the variable of the gamma function should be the dark pixel voltage $V_{DK}$ instead of thermister voltage $V_T$.

### 5.16.4 Data

#### *5.16.4.1 Input data*

The input data shall include the following time lines:

- Detector time lines after the nonlinearity correction (in Jy).
- Thermistor time lines after the deglitching (in V).
- Bias voltage flag indicating high or low bias.
- Time span (default 5 sec) for thermister timeline smoothing.
- A flag indicating whether the signal timeline of T1, T2, or both should be used in the correction.

#### *5.16.4.2 Input Calibration Products*

There shall be two sets of input calibration products, one for Photometer Pipeline and the other for the Spectrometer Pipeline. For the Photometer Pipeline, it shall contain the following parameters:

- The nominal detector array thermister signal $V_{T0}$ for all 6 thermisters (PS_T1, PS_T2, PM_T1, PM_T2, PL_T1, PL_T2).
- Tables of $A_i$ and $B_i$, which specify the gamma function, of all detectors. These parameters are different for different bias voltages. For each of the two nominal bias (low or high), there should be two sets of tables, one

for T1 and the other for T2. Altogether, there shall be four sets of tables (2 bias × 2 Ts). Note that for the high bias voltage, the thermisters are replaced by the dark pixels.

For the Spectrometer Pipeline, the parameters are:

- The nominal detector array thermister signal $V_{T0}$ for all 4 thermisters (SS_T1, SS_T2, SL_T1, SL_T2).
- Tables of $A_i$ and $B_i$, which specify the gamma function, of all detectors. These parameters are different for different bias voltages. For each of the two nominal bias (low or high), there should be two sets of tables, one for T1 and the other for T2. Altogether, there shall be four sets of tables (2 bias × 2 Ts).

#### *5.16.4.3 Output Data*

The output of the module will be detector time lines with the same structure and units as the input detector signal time lines.

### 5.16.5 Functional Requirements

**TCOR-FUN-010** The module shall operate on timelines in the Input Data list.

**TCOR-FUN-020** The module shall subtract the correlated noise in the detector time lines due to the array bath temperature drift.

**TCOR-FUN-030** The module shall be based on the empirical approach exploiting the tight correlation between the detector signals and the thermister signals, as described in AD03.

**TCOR-FUN-040** The modification presented in AD04, which subtracts the noise due to temperature drift after the non-linearity correction, has the advantage of being independent of the optical power. The module shall run with this algorithm.

**TCOR-FUN-050** The module shall be able to do the correction for detectors in each array using either single thermister (T1 or T2) time line, or both T1 and T2 time lines.

**TCOR-FUN-060** The module shall be able to do the correction for observations data taking at either one of the two nominal bias voltages. For the high bias voltage observations, the correction shall be done using the dark pixels instead of thermisters.

**TCOR-FUN-070** When using the single T1 or T2 voltage time line, the time line is first binned into bins of $\Delta t$ sec time span and averaged within each bin, where $\Delta t$ is a parameter. This filters out the high frequency variations that may not be related to the temperature fluctuations. Then a SPLINE fit to the smoothed timeline is carried out. This fit, denoted as $\underline{V_T}(t)$, specifies the bath temperature drift with time. The next step is, for each detector in a given array, to derive the power drift due to the temperature drift (AD04):
$\underline{Q_{T,i}}(t) = A_i \times (\underline{V_T}(t) - V_{T0}) + 0.5 \times B_i \times (\underline{V_T}(t) - V_{T0})^2$.
The parameters $A_i$ and $B_i$, are taken from the calibration table corresponding to the given bias voltage (low or high) and the thermister (T1 or T2). Subtract $\underline{Q_{T,i}}$ from the detector signal time line. The results are the temperature drift corrected signals.

**TCOR-FUN-090** When using both T1 and T2 time lines to do the correction, the above procedure is carried out for each detector channel twice using T1 and T2 time lines, respectively. The final estimate of the power drift is the average of the two: $\underline{Q_{T,i}}(t) = (\underline{Q_{T1,i}}(t) + \underline{Q_{T2,i}}(t))/2$

**TCOR-FUN-100** The module shall take as parameters the following:

- A switch to specify whether T1, T2, or T1+T2 thermister timeline shall be used. If not provided, the default is T1 only.

- Δt which specifies the time span of the binning and averaging of the thermister timeline before the SPLINE fit. If not provided, the default value is Δt = 5 sec.

## 5.17 Subtract Operating Point Voltage

## 5.18 Correct Bolometer Time Response

## 5.19 Nonlinearity Correction and Flux Density Conversion

This module, as part of the SPIRE empirical pipeline, converts a bolometer voltage reading in volts to a flux density in Jy. It applies to both photometer and spectrometer. It is based on two logical steps: (1) the nonlinear response of a bolometer detector to optical load is linearized based on an input calibration table and (2) the linearized signal is subsequently multiplied by a conversion factor to derive a flux density in Jy. For each of the photometer bands, the result equals the in-beam flux density at the filter reference wavelength for a flat (i.e., $\nu * f_\nu$ = const) continuum source; for the spectrometer, the result is consistent with the in-beam flux densities in Jy at TBD1 and TBD2 microns for a power-law continuum of a spectral index alpha (TBD3) over SSW and SLW, respectively.

In the photometer pipelines, this module is implemented after the module that removes electrical crosstalk; in the spectrometer pipeline, it is implemented just prior to the generation of an interferogram.

The algorithms implemented in this module are based on the formulae given in the reference AD05 (cf. Section 5.4): The incremental detector responsivity, dV/dQ, is a function of the total (bias + optical power) detector voltage V_tot, where Q is the optical power in Watts from the target source. Replacing Q with S, the properly defined source flux density in Jy (cf. AD05, Section 7) , one can write dS/dV_tot = f(V_tot) , which can be approximated by a 2$^{nd}$-order polynomial: K1 + K2*V_tot + K3*V_tot^2, where K1, K2, and K3 are constants for a given instrument/detector configuration. Thus, S equals an integral of the function f(V_tot) from V_o to V_(o+s) , where V_o and V_(o+s) are the bolometer voltage readings of the telescope and (telescope + source), respectively. PCAL observations on various backgrounds (e.g., between V_o and V_max) will determine the values of K1, K2 and K3. As a result, a calibration table can be constructed so that it contains values of V_o, K1, K2 and K3 (and their uncertainties) for each and every detector channel. This table converts a voltage reading V to a relative in-beam flux density S. The subsequent absolute flux calibration of S into Jy is done by multiplying S by a numeric factor, which is given in another calibration table, derived from observations of SPIRE flux standards.

There are a few technical notes worth pointing out: (i) This module applies to both photometer and spectrometer pipelines. (ii) In the case of the pipeline for jiggle observations, we assume that the signal demodulation per chop cycle is done in another module (i.e., unlike what is currently stated in AD04, Section 6.3). (iii) In the spectrometer case, the removal of the reference interferogram of (telescope+SCAL) occurs at a later stage in the pipeline, thus V_o = 0 (*To Be Confirmed*). (iv) Strictly speaking, f(V) should be measured at a fixed detector base temperature T_0 (TBD). (v) In addition to the SPIRE nominal operation mode, there will be at least a bright-source mode with the detectors under a different set of biasing conditions. So the calibration tables for this module may contain L (L>=1) layers with each layer representing one of the L possible detector biasing conditions. (vi) This module is not applicable to the model-based pipelines, in which the responsivity nonlinearity is addressed as a part of the bolometer physical model.

### 5.19.1 Module Owner

NHSC (Arnold Schwartz: implementation; Nanyao Lu: science requirements).

### 5.19.2 Others Contributing

TBD.

### 5.19.3 List of Applicable Documents

AD05 SPIRE Analogue Signal Chain and Photometer Detector Data Processing Pipeline (SPIRE-UCF-DOC-002890, Issue 4, 6 January 2008)
AD01 SPIRE Pipeline Description (SPIRE-RAL-DOC-002437)
AD06 SPIRE Spectrometer Pipeline Description (SPIRE-BSS-DOC-002966)

### 5.19.4 List of Reference Documents

RD01 SPIRE Data Products Specification (SPIRE-RAL-DOC-002005)

## 5.20

## 5.21 Module Description

The purpose of this module is to apply a correction for the nonlinear bolometer responsivity to a detector time line (RPDT or RSDT) and at the same time convert a detector signal to an in-beam source flux density in Jy. This is done via two calibration tables. The first calibration table ("SCalPhotNonLinCorr" for the photometer or "SCalSpecNonLinCorr" for the spectrometer) linearizes the voltages. The second calibration table ("SCalPhotUnittoAst" for the photometer or "SCalSpecUnitToAst" for the spectrometer) is then used to convert the results into in-beam flux densities in Jy.

## 5.22 Data

#### *5.22.1.1 Input Data*

| | | |
|---|---|---|
| **INPUT-010** | Detector signal time lines. | RPDT or RSDT |
| **INPUT-020** | The signals are quality tagged. | |
| **INPUT-030** | The detector biasing flag is provided if multiple biases are used for SPIRE (e.g., bright source observing mode). | Integer flag:<br>BiasFlag = 1,2,..,Lp |

#### *5.22.1.2 Input Calibration Data*

| | | |
|---|---|---|
| **INCAL-010** | A (Mp*8)*Lp table for the photometer arrays that tabulates V_o, V_o_err, K1, K1_err, K2, K2_err, K3, and K3_err for each of the Mp detector channels. Those inoperative channels may be left out of the table or assigned with a nominal value in the table (TBD). Lp = the number of possible detector biasing conditions in use for the SPIRE photometer mode. For example, Lp = 1 if only one detector biasing condition is used for the mission. Lp = 2 if there is an additional bright-source | Filename:<br>SCalPhotNonLinCorr |

| | | |
|---|---|---|
| | observing mode. The input flag "BiasFlag" determines which layer of the calibration table to use. The table header (or meta data) should contain the essential data of the table, including the version number, date of creation, and valid voltage limits. | |
| **INCAL-020** | A (Ms*8)*Ls table for the spectrometer arrays that tabulates V_o, V_o_err, K1, K1_err, K2, K2_err, K3, and K3_err for each of the Ms detector channels. Those inoperative channels may be left out of the table or assigned with a nominal value in the table (TBD). Lp = the number of possible detector biasing conditions in use for the SPIRE photometer mode. For example, Ls = 1 if only one detector biasing condition is used for the mission. Ls = 2 if there is an additional bright-source observing mode. The input flag "BiasFlag" determines which layer of the calibration table to use. . The table header (or meta data) should contain the essential data of the table, including the version number, date of creation, and valid voltage limits. | Filename: SCalSpecNonLinCorr |
| **INCAL-030** | A (Mp*2)*Lp table for the photometer arrays that tabulates the multiplicative absolute flux calibration factor and its uncertainty for each of Mp detector channels. Those inoperative channels may be left out of the table or assigned with a nominal value in the table (TBD). Lp = the number of possible detector biasing conditions in use for the SPIRE photometer mode. For example, Lp = 1 if only one detector biasing condition is used for the mission. Lp = 2 if there is an additional bright-source observing mode. The input flag "BiasFlag" determines which layer of the calibration table to use. . The table header (or meta data) should contain the essential data of the table, including the version number, date of creation, and valid flux density limits. | Filename: SCalPhotUnitToAst |
| **INCAL-040** | A (Mp*2)*Ls table for the spectrometer arrays that tabulates the multiplicative absolute flux calibration factor and its uncertainty for each of Mp detector channels. Those inoperative channels may be left out of the table or assigned with a nominal value in the table (TBD). Lp = the number of possible detector biasing conditions in use for the SPIRE photometer mode. For example, Ls = 1 if only one detector biasing condition is used for the mission. Ls = 2 if there is an additional bright-source observing mode. The input flag "BiasFlag" determines which layer of the calibration table to use. . The table header (or meta data) should contain the essential data of the table, including the version number, date of creation, and valid flux density limits. | Filename: SCalSpecUnitToAst |

#### *5.22.1.3 Output Data*

| | | |
|---|---|---|
| **OUTPUT-010** | The output data are the detector time lines of in-beam flux densities in units of Jy (per beam). | |

### 5.22.2 Functional Requirements

| | | |
|---|---|---|
| **NLCOR-FUN-010** | The module shall operate on detector time lines from a context of SPIRE photometer or spectrometer products. | |
| **NLCOR-FUN-020** | The module shall be applicable to both photometer and spectrometer data. | |
| **NLCOR-FUN-030** | The module should perform an integration between V_o and the input voltage using the polynomial coefficients in the calibration table INCAL-010 or INCAL-020 if the observed detector voltage is between the calibrated voltage limits of the table. | |
| **NLCOR-FUN-040** | If the input voltage is outside the calibrated voltage limits of the calibration table INCAL-010 or INCAL-020, an extrapolation of some form (model-based? -- TBD) shall be used. And a flag should be set in the output data. | Integer Flag: VoltgeOutBound = 1 |
| **NLCOR-FUN-050** | The module should calculate errors associated with the calculations in this module and appropriately propagate it to the detector signal error time lines. | |
| **NLCOR-FUN-060** | The module should be able to detect and treat properly flagged or masked detector channels. | |
| **NLCOR-FUN-070** | The module should be able to select the correct “layer” in the calibration tables INCAL-010/INCAL-030 or INCAL-020/INCAL-040 based on the input “BiasFlag” parameter. | |
| **NLCOR-FUN-080** | The module should appropriately indicate in the output data that this module has been performed on the data and the version numbers of this module and its associated calibration tables. | |

# 6. PHOTOMETER PROCESSING

## 6.1 Demodulation

### 6.1.1 Module Owner

CEA Saclay (René Gastaud responsible).

### 6.1.2 Others Contributing

Currently none.

### 6.1.3 Module Description

This module is one of the processing steps used to process data from the detectors. It shall compute the amplitude modulation when the Beam Steering Mirror chops the input signal. This will help to compute the flux of each pixel on the sky. There is one value per chopper cycle for each bolometer.

### 6.1.4 Data

#### *6.1.4.1 Input data*

There are three input products: Photometer Detector Timeline, Spire Pointing Product, Chop and Jiggle Timeline.

- PDT: contains the signal timeline
- CJT: contains the jiggle id with the start time and end time, and the chop time for each chopper motion
- SPP: this is not a product but a context. It includes a method for obtaining coordinates (ra, dec) for a given time and a given pixel.

#### *6.1.4.2 Output Data*

The output data will be in the form of an Averaged Demodulated Detector Timeline. It will contain the timeline of the demodulated values, with the sky coordinates (ra, dec), for both On and Off positions of the chopper, and the associated errors. There is one value per chopper cycle.

#### *6.1.4.3 Functional Requirements*

| | |
|---|---|
| **Demodulate-FUN-010** | The module shall process data from a single PDT |
| **Demodulate-FUN-020** | The module shall compute the demodulation using the best method (SPIRE-UCF-DOC-002890, The SPIRE Analogue Signal Chain and Photometer Detector Data Processing Pipeline Matt Griffin). |
| **Demodulate-FUN-030** | The calculated quantities are:<br>the modulation, the position On and Off for each bolometer. This will be stored in a DDT. |
| **Demodulate-FUN-040** | The module shall report missing steps. |

## 6.2 Second Level Deglitching (Chopped Data)

### 6.2.1 Module Owner

CEA Saclay (René Gastaud responsible).

### 6.2.2 Others Contributing

Currently none.

### 6.2.3 Module Description

This module applies only to photometer data modulated by the chopper. It shall be called before the demodulation task. For each timeline of each bolometer, it labels the samples “ON” , “OFF”, or moving. Then it computes a trend by linear regression for each jiggle state, both for “On” samples and “OFF” samples. Samples which are too far away from the trend are labelled “glitches”.

#### *6.2.3.1 Input data*

There are three input products: Photometer Detector Timeline, Chop and Jiggle Timeline.

- PDT: contains the signal timeline
- CJT: contains the chop time for each chopper motion

#### *6.2.3.2 Output Data*

The output data will be the input Data, Photometer Detector Timeline, with the mask updated for each bolometer.

#### *6.2.3.3 Functional Requirements*

| | |
|---|---|
| **secDeglitch-FUN-010** | The module shall process data from a single PDT |
| **secDeglitch -FUN-020** | The module shall label the samples. The labels are stored in the mask |
| **secDeglitch -FUN-030** | The module shall fill a quality flag. |
| **secDeglitch -FUN-040** | The output is the same PDT, with updated mask, and quality flag. |

## 6.3 Average Nod Cycles

### 6.3.1 Module Owner

CEA Saclay (René Gastaud responsible).

### 6.3.2 Others Contributing

Currently none.

### 6.3.3 Module Description

This module applies only to photometer data modulated by the chopper. It shall be called after the demodulation task. It computes for each bolometer the mean for each jiggle state (jiggle positions constant). The input is a Demodulated Detector Timeline with one value per chopper cycle and the output is Averaged Demodulated Detector Timeline with one value per jiggle state.

### 6.3.4 Data

#### *6.3.4.1 Input data*

- There is one input product: Demodulated Detector Timeline with one value per chopper cycle.

#### *6.3.4.2 Output Data*

The output data will be in the form of an Averaged Demodulated Detector Timeline with one value per jiggle state.

#### *6.3.4.3 Functional Requirements*

| | |
|---|---|
| **JiggAverage-FUN-010** | The module shall process data from a single DDT. |
| **JiggAverage-FUN-020** | The module shall compute the mean of the signal, the mean of the position, update the errors, for each jiggle states. |

**JiggAverage-FUN-030** The output is a single ADT.

## 6.4 De-Nod

### 6.4.1 Module Owner

Davide Rizzo (Imperial College)

### 6.4.2 Others Contributing

### 6.4.3 Module Description

This is the first pipeline step that cannot work on a single building block of detector data but instead must work on an observation. Each averaged pixel output at a nod position will contain the source flux minus a reference flux for that nod position. This module denodds the data to return the source flux in each pixel. If there are several visits to the same nod position, these will be averaged together.

### 6.4.4 Data

#### *6.4.4.1 Input data*

*6.4.4.2 The input data is a series of Averaged Demodulated Timelines (ADTs), one for each nod position.*

#### *6.4.4.3 Input Calibration Products*

There are no input calibration products. All the data necessary for this module is included in the input ADT products.

#### *6.4.4.4 Output Data*

The first task of this module will output a Pointed Photometer Product (PPP) containing the results of denodding for a single nod cycle (A-B or A-B-B-A, where A and B are the two nod positions).
The second task of the module will take one or more PPP as input and will average them, producing an Averaged Pointed Photometer Product (APPP) as output.

### 6.4.5 Functional Requirements

**DENOD-FUN-010** The module shall process data from one, two or four Averaged Demodulated Timelines (ADTs)

**DENOD-FUN-020** The module shall produce one or more Pointed Photometer Products (PPPs) as intermediate output and one Averaged Pointed Photometer Product (APPP) as final output.

| | | |
|---|---|---|
| **DENOD-FUN-030** | If only one ADT is given as input, the module will not perform any denodding but just average the signal for each detector and jiggle position | |
| **DENOD-FUN-040** | If two ADTs are given as input (A-B cycle) the signal will be given by the formula<br>*Signal = A - B* | |
| **DENOD-FUN-050** | If four ADTs are given as input (A1-B2-B3-A4 cycle) the signal | |

| | | |
|---|---|---|
| | will be given by the formula<br>Signal = ½ ((A1 - B3) + (A4 - B2) | |
| **DENOD-FUN-055** | Incomplete nod cycles (ABB) are dropped. When they occur a quality control flag is raised. | |
| **DENOD-FUN-060** | In case of more than one visit to the same nod position, the resulting signals will be averaged together. This will be done by computing a mean (default) or a median for each detector. | |
| **DENOD-FUN-070** | The averaging step will optionally allow the rejection of outliers. | |

## 6.5 Derive Point Source Flux Density and Position

### 6.5.1 Module Owner

Davide Rizzo (Imperial College)

### 6.5.2 Others Contributing

### 6.5.3 Module Description

This is the last module in the seven-point jiggle pipeline (POF 2). Its purpose is to give the flux and position of the point source being observed.
As an intermediate step, the module will also try to fit a functional representation of the PSF (currently a 2D symmetric Gaussian) to all pixels.

### 6.5.4 Data

#### *6.5.4.1 Input data*

*6.5.4.2 The input data is an Averaged Pointed Photometer Product (APPP) containing the position and background-subtracted signal for each pixel and each jiggle position.*

#### *6.5.4.3 Input Calibration Products*

This module will have to convert the flux of the source into astronomical units. A calibration product will have to be provided with the appropriate conversion factors.
This calibration product is still TBD.

#### *6.5.4.4 Output Data*

The first task of this module will output a Jiggle Point Source Fit Product (JPSFP) containing the results of the Gaussian fit on all pixels. This product will be fed into the second task, which will produce a Jiggle Photometer Product (JPP) with the position and flux of the source computed on each of the three arrays.

### 6.5.5 Functional Requirements

**SRCPOW-FUN-010** The module shall only process data from seven-point jiggle observations (POF 2)

**SRCPOW-FUN-020** The module shall process data from a single Averaged Pointed

Photometer Product (APPP)

| | | |
|---|---|---|
| **SRCPOW-FUN-030** | The module shall try to fit the signal from each pixel with a functional representation of the PSF, and output the resulting parameters if the fit is successful. | |
| **SRCPOW-FUN-031** | The module shall try a fit assuming that the source is exactly centred on a pixel, and then try again with a series of off-centre positions. The number and offsets of off-centre positions is TBD. | |
| **SRCPOW-FUN-040** | The module shall use a symmetric two-dimensional Gaussian to represent the PSF. | |
| **SRCPOW-FUN-050** | The module shall produce a final product containing the position and flux of the point source, with their errors, as calculated from each array. | |
| **SRCPOW-FUN-060** | The module shall provide, for each pixel, the sum of the signal on the seven jiggle points in addition to the fit parameters. | |
| **SRCPOW-FUN-070** | The module shall convert the source flux into astronomical units. | |

## 6.6 Mapmaking

This module is one of the final processing steps used to process time ordered series (TODs) produced by the instrument in scan mode (POF5) to provide maps of the 3 bolometer arrays. As both SPIRE and PACS are able to produce TODs from their observations and these should be as far as possible instrument independent, it is expected that this software will be able to deal with data from both instruments although some instrument specific processing may be needed.

### 6.6.1 Module Owner

Imperial College (Pierre Chanial responsible)

### 6.6.2 Others Contributing

None.

### 6.6.3 Module Description

The purpose of this module is to process a scan mode timeline into two-dimension sky maps for the 3 bolometer arrays.

### 6.6.4 Data

#### *6.6.4.1 Input data*

The following requirements are placed on the input data:

- Input data has been observed using the POF5 observation template.
- Input data is in the form of a collection of scan mode timelines and is in a Photometer Detector Timeline type product
- Input data is in the form of calibrated data samples with instrumental effects removed including array-level thermal drifts but excluding detector-level correlated noise.

- Input data has good astrometric calibration (including header information to convert between sky coordinates and map coordinates)
- Input data has good flux calibration (including a statement of the units)
- Input data has good quality flags.
- Datacubes include information on which detector timeseries each data sample is from (and which timestamp)
- Input data samples from each building block are evenly spaced in time.

#### *6.6.4.2 Input calibration files*

- This module also requires the calibration product SCalPhotInvNtt. This is the first row of the inverse time-time noise correlation matrix as a calibration product. It can be derived from scan map observations and is calculated using $CODE

#### *6.6.4.3 Output Data*

The output data will be in the form of a SPIRE Photometer Scanmap Product as described in requirements **SCNMAP-FUN-070 and SCNMAP-FUN-071.**

### 6.6.5 Functional Requirements

**SCNMAP-FUN-010** The module shall operate on data from a Context of Spire Photometer Scan Products.

**SCNMAP-FUN-020** The module shall provide an instrument-independent naïve mapper

**SCNMAP-FUN-021** The module shall provide an instrument-independent MADmap mapper, whose reference implementation in C is http://crd.lbl.gov/~cmc/MADmap/doc/

**SCNMAP-FUN-030** The module shall provide an interface between the Spire products and the naïve mapper.

**SCNMAP-FUN-031** The module shall provide an interface between the Spire products and the madmap mapper.

**SCNMAP-FUN-040** The module shall provide a submodule to derive the power spectrum density of a timeline

**SCNMAP-FUN-050** The module shall provide a submodule to derive the first row of the inverse time-time noise correlation matrix from the power spectrum density of a timeline

**SCNMAP-FUN-060** The naïve mapper shall take as parameters the following:

- *an output map resolution, otherwise default values shall be provided*
- *map coordinates timelines for each detector and each building block*
- *signal and mask timelines for each detector and each building block*

**SCNMAP-FUN-061** The MADmap mapper shall take as parameters the following:

- *an output map resolution, otherwise default values shall be provided*
- *map coordinates timelines for each detector, array and building block*
- *signal and mask timelines for each detector, array and building block*
- *the first row of the inverse time-time noise correlation matrix as a calibration product*
- *maximum number of iterations in the Preconjugate Gradient Method (PCG), otherwise a default value shall be provided*
- *maximum relative error allowed in PCG routine, otherwise a default value shall be provided*

| | |
|---|---|
| **SCNMAP-FUN-070** | The module using the naïve mapper shall produce a Spire Photometer Scan Map Product as output.<br>This product shall contain at least the following information for each bolometer array:<br>• *Estimated surface brightness*<br>• *Estimated error*<br>• *Unit*<br>• *Coverage*<br>• *Astrometry headers to convert map and sky coordinates* |
| **SCNMAP-FUN-071** | The module using the MADmap mapper shall produce a Spire Photometer Scan Map Product as output.<br>This product shall contain at least the following information for each bolometer array:<br>• *Estimated surface brightness*<br>• *Estimated error*<br>• *Unit*<br>• *Coverage*<br>• *Astrometry headers to convert map and sky coordinates* |
| **SCNMAP-FUN-080** | The naïve mapper shall be able to cope with observations exceeding the computer RAM memory. |
| **SCNMAP-FUN-081** | The madmap mapper shall be able to cope with observations exceeding the computer RAM memory. |

### 6.6.6 Performance Requirements

The following modifications and additional assumptions are applied to the Global Performance Requirement **SPIRE-PERF-020:**

| | |
|---|---|
| **SCNMAP-PERF-021** | The naïve mapper shall be able to process a 'typical' map on a standard desktop machine in less than 10 mins (TBC)<br>*A typical map is assumed to be approximately 0.5 by 0.5 degrees* |
| **SCNMAP-PERF-022** | The madmap mapper be able to process a 'typical' map on a standard desktop machine in less than 60 mins (TBC)<br>*A typical map is assumed to be approximately 0.5 by 0.5 degrees* |

# 7. SPECTROMETER PROCESSING

## 7.1 Clipping Correction

### 7.1.1 Module Owner

LAM (Dominique Benielli)

### 7.1.2 Module description

The purpose of the clipping module is to reconstruct the truncated part of the spectrometer detector timeline. The truncated samples have already been flagged by the *check ADC flags and Truncation* module. Truncated parts are reconstructed by using a sinc interpolation.

### 7.1.3 Data

#### *7.1.3.1 Input Data*

The input data will be in the form of a spectrometer detector timeline.

#### *7.1.3.2 Input Calibration Data*

No calibration data are needed.

#### *7.1.3.3 Output Data*

The output data will be in the form of a spectrometer detector timeline.

### 7.1.4 Functional Requirements

| | | |
|---|---|---|
| **CLIPPING-FUN-010** | The module shall operate on data from a spectrometer detector timeline. | |
| **CLIPPING-FUN-020** | Clipped samples shall be given by reading the corresponding bit of the mask timeline. Clipped samples are the ones flagged by the *check ADC flags and Truncation* module (section 7). | |
| **CLIPPING-FUN-030** | The module shall reconstruct flagged data by using a sinc interpolation. | |

## 7.2 Time Domain Phase Correction

### 7.2.1 Module Owner

Blue Sky Spectroscopy (Trevor Fulton).

### 7.2.2 Module Description

The purpose of this module is to adjust the spectrometer detector samples to account for the time delay induced by the detector read-out electronics and the thermal response of the detectors.

### 7.2.3 Data

#### *7.2.3.1 Input Data*

| | | |
|---|---|---|
| **TDPHASECORR-INP-010** | Input science data has been observed using an astronomical observation template AOT. | |
| **TDPHASECORR-INP-020** | Input science data shall be in the form of a Spectrometer Detector Timeline (SDT) product. | SDT |
| **TDPHASECORR-INP-021** | Input science data shall have its "type" metadata parameter set to "SDT". | |
| **TDPHASECORR-INP-022** | Input science data shall have its "commandedResolution" metadata parameter set to "HR", "MR", or "LR". | |

| | | |
|---|---|---|
| **TDPHASECORR-INP-024** | Detectors denoted as dead or noisy have had their respective sample masks set for all of their samples. | |

#### *7.2.3.2 Input Calibration Data*

| | | |
|---|---|---|
| **TDPHASECORR-CAL-010** | A calibration product that identifies illuminated detectors and unconnected channels shall be provided. | |
| **TDPHASECORR-CAL-020** | An input calibration data that contains the characteristics for the spectrometer read-out electronics shall be provided. | |
| **TDPHASECORR-CAL-021** | This calibration product shall contain the values of the resistors and capacitors in the spectrometer read-out electronics. | |
| **TDPHASECORR-CAL-030** | An input calibration data that contains the thermal time constant for each spectrometer detector shall be provided. | |

#### *7.2.3.3 Output Data*

| | | |
|---|---|---|
| **TDPHASECORR-OUT-020** | Output science data shall be in the form of a Spectrometer Detector Timeline (SDT) product. | SDT |
| **TDPHASECORR-OUT-021** | The "Type" metadata parameter in the output SDT product shall be set to "SDT”. | |
| **TDPHASECORR-OUT-022** | The "commandedResolution" metadata parameter in the output SDT product shall be set to the same value is was the case for the input SDT product. | |
| **TDPHASECORR-OUT-024** | Detectors denoted as dead or noisy in the input SDT product will likewise be denoted as such in the output SDT product and will have their respective sample masks set for all of their samples. | |

### 7.2.4 Functional Requirements

| | | |
|---|---|---|
| **TDPHASECORR-FUN-010** | The module shall compute the overall time phase imparted on the detector timelines based on the information contained in the input calibration products. | |
| **TDPHASECORR-FUN-020** | The module shall modify the signal samples of each spectrometer detector by convolution with the inverse transform of the time phase derived from the information in the input calibration products. | |

## 7.3 Interferogram Creation

### 7.3.1 Module Owner

Blue Sky Spectroscopy (Trevor Fulton).

### 7.3.2 Module Description

A typical SPIRE spectrometer observation consists of a series of scans of the spectrometer mechanism while the instrument is pointed at a given target. In the SPIRE spectrometer, the sampling of spectrometer detectors and the spectrometer mechanism is decoupled; the two subsystems are sampled at different rates and at different times. The purpose of this module is to combine the spectrometer detector timelines and spectrometer mechanism timeline into a set of interferograms whose samples are equidistant for a given SPIRE spectrometer observation..

### 7.3.3 Data

#### *7.3.3.1 Input Data*

| | | |
|---|---|---|
| **CREATEIFGM-INP-010** | Input science data has been observed using an astronomical observation template AOT. | |
| **CREATEIFGM-INP-015** | Input science data shall be from a single SPIRE spectrometer building block. | |
| **CREATEIFGM-INP-016** | The OBSID metadata parameter for all input science products shall be the same. | |
| **CREATEIFGM-INP-017** | The BBID metadata parameter for all input science products shall be the same. | |
| **CREATEIFGM-INP-020** | One of the input science data products shall be a Spectrometer Detector Timeline product (SDT). | SDT |
| **CREATEIFGM-INP-021** | The input SDT product shall have its "type" metadata parameter set to "SDT". | |
| **CREATEIFGM-INP-022** | The input SDT product shall have its "commandedResolution" metadata parameter set to "HR", "MR", or "LR". | |
| **CREATEIFGM-INP-024** | Detectors denoted as dead or noisy shall have had their respective sample masks set for all of their samples in the input SDT product. | |
| **CREATEIFGM-INP-030** | One of the input science data products shall be a Spectrometer Mechanism Timeline product (SMECT). | SMECT |
| **CREATEIFGM-INP-031** | The input SMECT product shall contain timelines for the Spectrometer Mechanism coarse optical encoder. | |
| **CREATEIFGM-INP-032** | The input SMECT product shall contain a timeline for the Spectrometer Mechanism fine optical encoder. | |
| **CREATEIFGM-INP-033** | The input SMECT product shall contain a timeline for the Spectrometer Mechanism LVDT. | |
| **CREATEIFGM-INP-040** | One of the input science data products shall be a Nominal Housekeeping Timeline product (NHKT). | NHKT |
| **CREATEIFGM-INP-041** | The input NHKT product shall contain a timeline for the SCANS nominal housekeeping parameter. | |
| **CREATEIFGM-INP-042** | The input NHKT product shall contain a timeline for the SCANSTART nominal housekeeping parameter. | |
| **CREATEIFGM-INP-043** | The input NHKT product shall contain a timeline for the SCANSEND nominal housekeeping parameter. | |

| | | |
|---|---|---|
| **CREATEIFGM-INP-050** | One of the input science data products shall be a SPIRE Pointing product (SPP). | SPP |
| **CREATEIFGM-INP-051** | The input SPP product shall contain a timeline of the SPIRE pointing. | |

### *7.3.3.2 Input Calibration Products*

| | | |
|---|---|---|
| **CREATEIFGM-CAL-010** | A calibration product that identifies illuminated detectors and unconnected channels shall be provided. | |
| **CREATEIFGM-CAL-020** | A calibration product that contains the value of the mechanism optical encoder and LVDT at the position of zero path difference shall be provided. | |
| **CREATEIFGM-CAL-030** | A calibration product that contains the scale factor to be applied to convert a step in mechanical path difference to a step in optical path difference shall be provided. | |
| **CREATEIFGM-CAL-040** | A calibration product that contains the sample time offsets for each input data product representing the difference between the actual sample time and the frametime shall be provided. | |

### *7.3.3.3 Output Data*

| | | |
|---|---|---|
| **CREATEIFGM-OUT-010** | The output science product shall be in the form of a Spectrometer Detector Interferogam (SDI) product. | SDI |
| **CREATEIFGM-OUT-011** | The output science product shall have its “type”metadata parameter set to “SDI”. | |
| **CREATEIFGM-OUT-012** | The output SDI product shall have its “numScans” metadata parameter set to a value equivalent to the number of scans it contains. | |
| **CREATEIFGM-OUT-013** | The output SDI product shall have its “commandedResolution” metadata parameter set to the same value as that set in the input SDT product. | |
| **CREATEIFGM-OUT-014** | Detectors denoted as dead or noisy in the input SDT product will likewise be denoted as such in the output SDI product and will have their respective sample masks set for all of their samples. | |
| **CREATEIFGM-OUT-020** | The output SDI product shall contain a set of interferograms whose samples are regularly spaced in terms of OPD in units of centimeters (cm). | |
| **CREATEIFGM-OUT-021** | The interferograms in the the output SDI product shall contain a sample at the position of zero path difference (OPD = 0). | |
| **CREATEIFGM-OUT-030** | The RA and DEC metadata parameters in the output SDI product shall contain a single pointing value derived from the input SPIRE Pointing product. | |

### 7.3.4 Functional Requirements

| | | |
|---|---|---|
| **CREATEIFGM-FUN-010** | The mirror position samples for each scan in the input SMECT product shall be interpolated onto a grid of mirror positions that are regularly-spaced. | |
| **CREATEIFGM-FUN-011** | The regularly-spaced mirror position samples will be unique to each spectrometer detector. | |
| **CREATEIFGM-FUN-012** | The regularly-spaced mirror position samples for each spectrometer detector will be computed such that the OPD sampling intervals for all detectors are the same. | |
| **CREATEIFGM-FUN-013** | The regularly-spaced mirror position samples for each spectrometer detector shall be computed such that there will be a sample corresponding to the position of zero path difference for all detectors. | |
| **CREATEIFGM-FUN-020** | The signal samples for a given spectrometer shall be interpolated onto the OPD positions for that detector. | |
| **CREATEIFGM-FUN-030** | The pointing value affixed to the output SDI product shall be the time-averaged mean pointing value of the building block. | |

## 7.4 SCAL and Telescope Correction

### 7.4.1 Module owner

LAM (Dominique Benielli)

### 7.4.2 Other contributions

BlueSkySpectroscopy (Trevor Fulton, Peter-Davis Imhof)

### 7.4.3 Module description

This module applies source calibration and the telescope corrections to the input data.
In the case of bolometer theoretical model processing:

- the Scal module removes the Scal contribution from the spectrometer detector interferogram data.
- the Telescope module removes the telescope contribution from the spectrometer detector interferogram data

In the case of empirical bolometer processing (TDB), a blank sky record, containing the Scal and Telescope signals, is used as calibration data to apply the correction.
One of these two methods will be chosen on the basis of our ability to provide a good telescope model.

### 7.4.4 Data

#### *7.4.4.1 Input data*

The input data to be corrected is the spectrometer detector interferogram produced by the interferogram creation module. Units should be in picoWatt.

The Nominal House keeping and the spectrometer mechanism timeline are used by the theoretical processing model.

#### 7.4.4.2 Input calibration data

Two cases have to be considered :

- For theoretical bolometer processing the following calibration products, found in the calibration context, are needed:
  - for Scal sub-module : Scal relative spectrometer filter response (spec.scalRsrf), Scal emissivities (spec.scalEm), Scal optical phase (spec.nlp), optical encoder position at ZPD data (spec.smecZpd).
  - for Tel sub-module : Telescope calibration data (temperatures and emissivities – TBD), relative spectrometer response filter data.
- For empirical bolometer processing the only required calibration product is a spectrometer detector interferogram the blank sky.

These calibration products are produced by the module's author.

#### 7.4.4.3 Output Data

The output data is a spectrometer detector interferogram.

### 7.4.5 Functional requirements

- For theoretical model-based bolometer processing:

| | | |
|---|---|---|
| **SCAL-FUN-010** | The module shall operate on data in the form of a spectrometer detector interferogram. | |
| **SCAL-FUN-020** | The input data unit shall be in picoWatt | |
| **SCAL-FUN-030** | The module shall use the NHK and the SMEC Timeline to measure the source temperature at each ZPD location (i.e for each scan) | |
| **SCAL-FUN-040** | Source temperature fluctuations during each scan are assumed to be lower than 250 mK to build their Planck function | |
| **SCAL-FUN-041** | Bath temperature fluctuations during each scan are assumed to be lower than 30 mK to build its Planck function | |
| **SCAL-FUN-050** | The module shall compute the black body functions of the system (sources, bath) for each scan. | |
| **SCAL-FUN-051** | The module shall use the Scal emissivity functions and the source and bath black body functions to compute the theoretical grey body functions of the system (sources, bath) for each scan | |
| **SCAL-FUN-052** | The module shall use the scans names and the defined detectors in the input data to calculate a spectrometer detector spectrum of the Scal using the corresponding theoretical gray body functions | |
| **SCAL-FUN-053** | The module shall use the Scal relative spectrometer response function (RSRF) data to apply the RSRF correction to each detector in the Scal spectrum | |
| **SCAL-FUN-054** | The module shall use the Scal optical phase data to apply the optical phase correction to each detector in the Scal spectrum | |
| **SCAL-FUN-055** | The module shall perform an inverse Fourier transform on each dataset to build the Scal spectrometer detector interferogram | |
| **SCAL-FUN-060** | The Scal spectrometer detector interferogram is subtracted from the input data by applying a linear interpolation on the zero path difference (ZPD) positions. | |

(The following table is TBD) :

| | | |
|---|---|---|
| **TEL-FUN-010** | The module shall operate on data in the form of a spectrometer detector interferogram. | |
| **TEL-FUN-020** | The input data unit shall be in picoWatt | |
| **TEL-FUN-030** | The module shall use the Telescope calibration data to build a theoretical grey body function of the telescope | |
| **TEL-FUN-040** | The module shall use the RSRF data to apply the RSRF correction to each detector in the Telescope spectrum | |
| **TEL-FUN-050** | The module shall perform an inverse Fourier transform on each dataset to build the Telescope interferogram | |
| **TEL-FUN-060** | The Telescope interferogram is subtracted from the Scal corrected data by applying a linear interpolation on the ZPD positions | |

– For empirical bolometer processing: (TBD)

| | | |
|---|---|---|
| **SCALTEL-FUN-010** | The module shall operate on data in the form of a spectrometer detector interferogram | |
| **SCALTEL-FUN-020** | The module shall use spectrometer detector interferogram data of from blank sky to perform the SCALTEL correction | |
| **SCALTEL-FUN-021** | For each detector, the spectrometer detector interferogram data of blank sky are subtracted from the input data by applying a linear interpolation to ZPD positions | |

## 7.5 Interferogram Baseline Correction

### 7.5.1 Module Owner

Blue Sky Spectroscopy (Trevor Fulton).

### 7.5.2 Module Description

The purpose of this module is to remove the baseline from each of the interferograms of a SPIRE spectrometer observation.

### 7.5.3 Data

#### *7.5.3.1 Input Data*

| | | |
|---|---|---|
| **BASECORR-INP-010** | Input science data has been observed using an astronomical observation template AOT. | |
| **BASECORR-INP-020** | Input science data shall be in the form of a Spectrometer Detector Interferogram (SDI) product. | SDI |
| **BASECORR-INP-021** | Input science data shall have its "type" metadata parameter set to "SDI". | |
| **BASECORR-INP-022** | Input science data shall have its “numScans” metadata parameter set to a value equivalent to the number of scans it | |

| | | |
|---|---|---|
| | contains. | |
| **BASECORR-INP-023** | Input science data shall have its “commandedResolution” metadata parameter set to “HR”, “MR”,or “LR”. | |
| **BASECORR-INP-024** | Detectors denoted as dead or noisy have had their respective sample masks set for all of their samples. | |

#### *7.5.3.2 Input Calibration Products*

| | | |
|---|---|---|
| **BASECORR-CAL-010** | A calibration product that identifies illuminated detectors and unconnected channels shall be provided. | |

#### *7.5.3.3 Output Data*

| | | |
|---|---|---|
| **BASECORR-OUT-020** | Output science data shall be in the form of a Spectrometer Detector Interferogram (SDI) product. | SDI |
| **BASECORR-OUT-021** | Output science data shall have its "type" metadata parameter set to "SDI". | |
| **BASECORR-OUT-022** | Output science data shall have its "numScans" metadata parameter set to the same value as that set in the input SDI product. | |
| **BASECORR-OUT-023** | Output science data shall have its "commandedResolution" metadata parameter set to the same value as that set in the input SDI product. | |
| **BASECORR-OUT-024** | Detectors denoted as dead or noisy in the input SDI product will likewise be denoted as such in the output SDI product and will have their respective sample masks set for all of their samples. | |
| **BASECORR-OUT-030** | The module shall provide the option to produce an auxiliary output SDI data product that contains just the baselines that were removed from the interferograms in the input SDI product. | |

### 7.5.4 Functional Requirements

| | | |
|---|---|---|
| **BASECORR-FUN-010** | On an interferogram-by-interferogram basis, the module shall fit a position-dependent fourth-order polynomial. | |
| **BASECORR-FUN-020** | On an interferogram-by-interferogram basis, the module shall subtract from the input interferograms the fitted position-dependent baseline. | |

## 7.6 2nd Level Deglitching (Spectrometer)

### 7.6.1 Module Owner

Blue Sky Spectroscopy (Trevor Fulton).

### 7.6.2 Others Contributing

None

### 7.6.3 Module Description

The purpose of this module is to remove unwanted localized artefacts from the measured interferograms prior to transformation.

### 7.6.4 Data

#### *7.6.4.1 Input Data*

| | | |
|---|---|---|
| **IFGMDEGLITCH-INP-010** | Input science data has been observed using an astronomical observation template AOT. | |
| **IFGMDEGLITCH-INP-020** | Input science data shall be in the form of a Spectrometer Detector Interferogram (SDI) product. | SDI |
| **IFGMDEGLITCH-INP-021** | Input science data shall have its "type" metadata parameter set to "SDI". | |
| **IFGMDEGLITCH-INP-022** | Input science data shall have its "numScans" metadata parameter set to a value equivalent to the number of scans it contains. | |
| **IFGMDEGLITCH-INP-023** | Input science data shall have its "commandedResolution" metadata parameter set to "HR", "MR", or "LR". | |
| **IFGMDEGLITCH-INP-024** | Detectors denoted as dead or noisy have had their respective sample masks set for all of their samples in the input SDI product. | |
| **IFGMDEGLITCH-INP-030** | The sample positions for each interferogram for a given spectrometer detector in the input SDI product shall be the same. | |

#### *7.6.4.2 Input Calibration Data*

| | | |
|---|---|---|
| **IFGMDEGLITCH-CAL-010** | A calibration product that identifies illuminated detectors and unconnected channels shall be provided. | |

#### *7.6.4.3 Output Data*

| | | |
|---|---|---|
| **IFGMDEGLITCH-OUT-020** | Output science data shall be in the form of a Spectrometer Detector Interferogram (SDI) product. | SDI |
| **IFGMDEGLITCH-OUT-021** | Output science data shall have its "type" metadata parameter set to "SDI". | |
| **IFGMDEGLITCH-OUT-022** | Output science data shall have its "numScans" metadata parameter set to a value equivalent to the number of scans it contains. This value shall be the same as that in the input SDI product. | |
| **IFGMDEGLITCH-OUT-023** | Output science data shall have its "commandedResolution" metadata parameter set set to the same value as that in the input SDI product. | |
| **IFGMDEGLITCH-OUT-024** | Detectors denoted as dead or noisy in the input SDI product will likewise be denoted as such in the output | |

| | | |
|---|---|---|
| | SDI product and will have their respective sample masks set for all of their samples. | |
| **IFGMDEGLITCH-OUT-030** | Detector samples that have been identified as glitches by this module shall have the “DETECTED_GLITCH” bit set in their respective sample masks. | |
| **IFGMDEGLITCH-OUT-031** | Detector samples that have been modified by this module shall have the "UNCORRECTED_GLITCH" bit unset in their respective sample masks. | |
| **IFGMDEGLITCH-OUT-032** | Detector samples that have been identified as glitches by this module but were not subsequently modified shall have the "UNCORRECTED_GLITCH" bit set in their respective sample masks. | |

### 7.6.5 Functional Requirements

| | | |
|---|---|---|
| **IFGMDEGLITCH-FUN-010** | The module shall compute the standard deviation of the signal samples at each sample position across all scans for each illuminated detector in the input SDI product. | |
| **IFGMDEGLITCH-FUN-011** | The module shall compute the median of the standard deviations of the signal samples within a window for each illuminated detector in the input SDI product. | |
| **IFGMDEGLITCH-FUN-012** | The module shall identify samples as glitches those samples that exceed the glitch threshold. | |
| **IFGMDEGLITCH-FUN-013** | The module shall set the “DETECTED_GLITCH” and “UNCORRECTED_GLITCH” sample mask bits for those samples identified as glitches using the SpireMask interface. | |
| **IFGMDEGLITCH-FUN-020** | The module shall replace those samples identified as glitches by the mean signal sample derived from the non-glitch signal samples from the other scans at that sample position for that spectrometer detector. | |
| **IFGMDEGLITCH-FUN-021** | The module shall unset the “UNCORRECTED_GLITCH” sample mask bit for those samples that were corrected using the SpireMask interface. | |

## 7.7 Apodisation

### 7.7.1 Module Owner

Blue Sky Spectroscopy (Trevor Fulton).

### 7.7.2 Others Contributing

None

### 7.7.3 Module Description

The purpose of the Apodization task is to correct for and remove effects related to the instrumental line shape of the Fourier Transform spectrometer.

### 7.7.4 Data

#### *7.7.4.1 Input Data*

| | | |
|---|---|---|
| **APODIZE-INP-010** | Input science data has been observed using an astronomical observation template, AOT. | |
| **APODIZE-INP-020** | Input science data shall be in the form of a Spectrometer Detector Interferogram product. | SDI |
| **APODIZE-INP-021** | Input science data shall have its "type" metadata parameter set to "SDI". | |
| **APODIZE-INP-022** | Input science data shall have its "numScans" metadata parameter set to a value equivalent to the number of scans it contains. | |
| **APODIZE-INP-023** | Input science data shall have its "commandedResolution" metadata parameter set to "HR", "MR", or "LR". | |
| **APODIZE-INP-024** | Detectors denoted as dead or noisy have had their respective sample masks set for all of their samples. | |
| **APODIZE-INP-030** | If the "apodType" argument is set to "ds", each interferogram in the input SDI product for the illuminated detectors shall contain samples whose OPD range encompasses the position of ZPD. | |

#### *7.7.4.2 Calibration Data*

| | | |
|---|---|---|
| **APODIZE-CAL-010** | A calibration product that identifies illuminated detectors and unconnected channels shall be provided. | |

#### *7.7.4.3 Other Inputs*

| | | |
|---|---|---|
| **APODIZE-OTH-010** | The "apodType" input argument shall be set. to indicate whether singlesided or doublesided apodization is to be applied. | |
| **APODIZE-OTH-011** | The "apodType" argument shall be set to "ds" to apply doublesided apodization. | |
| **APODIZE-OTH-012** | The "apodType" argument shall be set to "ss" to apply singlesided apodization. | |

#### *7.7.4.4 Output Data*

| | | |
|---|---|---|
| **APODIZE-OUT-020** | Output science data shall be in the form of a Spectrometer Detector Interferogram (SDI) product. | SDI |
| **APODIZE-OUT-021** | Output science data shall have its "type" metadata parameter set to "SDI". | |
| **APODIZE-OUT-022** | Output science data shall have its "numScans" metadata parameter set to a value equivalent to the number of scans it contains. This value shall be the same as that in the input SDI | |

| | | |
|---|---|---|
| | product. | |
| **APODIZE-OUT-023** | Output science data shall have its "commandedResolution" metadata parameter set set to the same value as that in the input SDI product. | |
| **APODIZE-OUT-024** | Detectors denoted as dead or noisy in the input SDI product will likewise be denoted as such in the output SDI product and will have their respective sample masks set for all of their samples. | |
| **APODIZE-OUT-030** | If the "apodType" argument is set to "ds", the output SDI product will not overwrite the input SDI product. | |
| **APODIZE-OUT-031** | If the "apodType" argument is set to "ds", the interferograms in the output SDI product shall be truncated such that their sampled OPD values are symmetric about the position of ZPD. | |
| **APODIZE-OUT-040** | If the "apodType" keyword argument is set to "ss", the output SDI product will overwrite the input SDI product. | |
| **APODIZE-OUT-041** | If the "apodType" keyword argument is set to "ss", the interferograms in the output SDI product will be truncated such that their sampled OPD values are greater than or equal to the position of ZPD. | |

### 7.7.5 Functional Requirements

| | | |
|---|---|---|
| **APODIZE-FUN-010** | The module shall operate on data from a single spectrometer observation building block. | |
| **APODIZE-FUN-020** | The module shall be able to accommodate input interferograms that contain either an even number of samples or an odd number of samples. | |
| **APODIZE-FUN-030** | The module shall be able to operate on both singlesided and doublesided interferograms. | |
| **APODIZE-FUN-031** | If the singlesided option is chosen, the number of samples in each output interferogram($N_{Samples}$) shall be odd. | |
| **APODIZE-FUN-032** | If the doublesided option is chosen, the number of samples in each output interferogram ($N_{Samples}$) shall be even and shall be such that $N_{Samples}/2 + 1$ is odd. | |

## 7.8 Phase Correction

### 7.8.1 Module Owner

Blue Sky Spectroscopy (Trevor Fulton).

### 7.8.2 Module Description

The purpose of the module is to modify the input interferograms to correct for any asymmetries in the sampled signal. These asymmetries, if left uncorrected, lead to phase errors in the resultant spectrum.

### 7.8.3 Data

#### *7.8.3.1 Input Data*

| | | |
|---|---|---|
| **PHASECORR-INP-010** | Input science data has been observed using an astronomical observation template AOT. | |
| **PHASECORR-INP-011** | All input science data products shall be from a single SPIRE spectrometer building block. | |
| **PHASECORR-INP-012** | The "obsid" metadata parameter for all input science data products shall be the same. | |
| **PHASECORR-INP-013** | The "bbid" metadata parameter for all input science data products shall be the same. | |
| **PHASECORR-INP-020** | One of the input science data products shall be in the form of a Spectrometer Detector Interferogram (SDI) product. | SDI |
| **PHASECORR-INP-021** | The input SDI product shall have its "type" metadata parameter set to “SDI”. | |
| **PHASECORR-INP-022** | The input SDI product shall have its "numScans" metadata parameter set to a value equivalent to the number of scans it contains. | |
| **PHASECORR-INP-023** | The input SDI product shall have its "commandedResolution" metadata parameter set to "HR", "MR", or "LR". | |
| **PHASECORR-INP-024** | Detectors denoted as dead or noisy have had their respective sample masks set for all of their samples. | |
| **PHASECORR-INP-030** | One of the input science data products shall be in the form of a Spectrometer Detector Spectrum (SDS) product. | SDS |
| **PHASECORR-INP-031** | The input SDS product shall have its "type" metadata parameter set to “SDS”. | |
| **PHASECORR-INP-032** | The input SDS product shall have its "numScans" metadata parameter set to a value equivalent to the number of scans it contains. This value shall be equivalent to that set in the input SDI product. | |
| **PHASECORR-INP-033** | The input SDS product shall have its "commandedResolution" set to the same value as that in the input SDI product. | |
| **PHASECORR-INP-034** | Detectors denoted as dead or noisy have had their respective sample masks set for all of their samples. | |
| **PHASECORR-INP-035** | The input SDS product shall have its "actualResolution" set to the resolution of its spectra. This quantity shall be expressed in units of wavenumbers ($cm^{-1}$). | |
| **PHASECORR-INP-036** | The input SDS product shall contain a set of spectra derived from the doublesided portion of the interferograms in the input SDI product. | |

#### *7.8.3.2 Input Calibration Products*

| | | |
|---|---|---|
| **PHASECORR-CAL-010** | A calibration product that identifies illuminated detectors and unconnected channels shall be provided. | |
| **PHASECORR-CAL-020** | A calibration product that that contains the spectral band edges for each spectrometer detector shall be provided. | |
| **PHASECORR-CAL-030** | A calibration product that contains the position-dependent phase for each spectrometer detector shall be provided. | |

#### *7.8.3.3 Output Products*

| | | |
|---|---|---|
| **PHASECORR-OUT-020** | Output science data shall be in the form of a Spectrometer Detector Interferogram (SDI) product. | SDI |
| **PHASECORR-OUT-021** | Output science data shall have its "type" metadata parameter set to "SDI". | |
| **PHASECORR-OUT-022** | Output science data shall have its "numScans" metadata parameter set to a value equivalent to the number of scans it contains. This value shall be equivalent to that set in the input SDI product. | |
| **PHASECORR-OUT-023** | Output science data shall have its "commandedResolution" set to the same value as that in the input SDI product. | |
| **PHASECORR-OUT-024** | Detectors denoted as dead or noisy in the input SDI product will likewise be denoted as such in the output SDI product and will have their respective sample masks set for all of their samples. | |

### 7.8.4 Functional requirements

| | | |
|---|---|---|
| **PHASECORR-FUN-020** | The module shall compute the phase for each scan for each illuminated detector in the input science products. | |
| **PHASECORR-FUN-021** | The module shall compute a phase correction function for each scan for each illuminated detector in the input science products by way of a fourth-order polynomial fit to the measured in-band phase. | |
| **PHASECORR-FUN-030** | The module shall apply the computed phase correction function for a given scan for a given spectrometer detector to the input science data. | |
| **PHASECORR-FUN-031** | If the observation is of type low-resolution or medium-resolution, the phase correction function shall be applied to the science data in the input SDS product. | |
| **PHASECORR-FUN-032** | If the observation is of type low-resolution or medium-resolution, the output SDI product shall be derived from the inverse transform of the corrected SDS product. | |
| **PHASECORR-FUN-035** | If the observation is of type high-resolution the phase correction function shall be applied to the science data in the input SDS product. | |
| **PHASECORR-FUN-036** | If the observation is of type high-resolution, the inverse transform of the phase correction function shall be convolved with the science data in the input SDI product. | |

## 7.9 Fourier Transform

### 7.9.1 Module Owner

Blue Sky Spectroscopy (Trevor Fulton).

### 7.9.2 Others Contributing

None

### 7.9.3 Module Description

The purpose of this module is to create a set of spectra for each spectrometer detector channel for a given observation.

### 7.9.4 Data

#### *7.9.4.1 Input Science Data*

| | | |
|---|---|---|
| **FXFORM-INP-010** | Input science data has been observed using an astronomical observation template AOT. | |
| **FXFORM-INP-020** | Input science data shall be in the form of a Spectrometer Detector Interferogram (SDI) product. | SDI |
| **FXFORM-INP-021** | Input science data shall have its "type" metadata parameter set to "SDI". | |
| **FXFORM-INP-022** | Input science data shall have its "numScans" metadata parameter set to a value equivalent to the number of scans it contains. | |
| **FXFORM-INP-023** | Input science data shall have its "commandedResolution" metadata parameter set to "HR", "MR", or "LR". | |
| **FXFORM-INP-024** | Detectors denoted as dead or noisy have had their respective sample masks set for all of their samples. | |
| **FXFORM-INP-030** | The interferograms for all scans for a given spectrometer detector in the input SDI product shall contain the same number of samples and shall be sampled on a common set of OPDs. | |
| **FXFORM-INP-031** | The sampling interval for all interferograms in the input SDI product shall be the same. | |
| **FXFORM-INP-032** | Every interferogram in the input SDI product shall each contain a sample at the position of zero path difference (OPD = 0). | |
| **FXFORM-INP-033** | If the "ftType" argument is set to "ds", each interferogram in the input SDI product for the illuminated detectors shall contain samples whose OPD range encompasses the position of ZPD. | |

#### *7.9.4.2 Input Calibration Data*

| | | |
|---|---|---|
| **FXFORM-CAL-010** | A calibration product that identifies illuminated detectors and unconnected channels shall be provided. | |

#### *7.9.4.3 Other Inputs*

| | | |
|---|---|---|
| **FXFORM-OTH-010** | The "ftType" input argument shall be set. to indicate whether a singlesided or doublesided transformation is to be applied. | |
| **FXFORM-OTH-011** | The "ftType" argument shall be set to "ds" to apply a doublesided transformation. | |
| **FXFORM-OTH-011** | The "ftType" argument shall be set to "ss" to apply a singlesided transformation. | |

#### *7.9.4.4 Output Data*

| | | |
|---|---|---|
| **FXFORM-OUT-020** | The output science product shall be in the form of a Spectrometer Detector Spectrum (SDS) product. | SDS |
| **FXFORM-OUT-021** | The output SDS shall have its “type” metadata parameter set to “SDS”. | |
| **FXFORM-OUT-022** | The output SDS shall have its "numScans" metadata parameter set to a value equivalent to the number of scans it contains. This number shall be equivalent to the number of scans in the input SDI product. | |
| **FXFORM-OUT-023** | The output SDS product shall have its "commandedResolution" metadata parameter set to the same value as that in the input SDI product. | |
| **FXFORM-OUT-024** | Detectors denoted as dead or noisy in the input SDI product will likewise be denoted as such in the output SDS product and will have their respective sample masks set for all of their samples. | |
| **FXFORM-OUT-025** | The output SDS product shall have its "actualResolution" metadata parameter set to the resolution of its spectra. This quantity shall be expressed in units of wavenumbers ($cm^{-1}$). | |
| **FXFORM-OUT-030** | The spectral resolution of the spectra in the output SDS product shall be the same for all scans and all spectrometer detectors. | |
| **FXFORM-OUT-031** | The Nyquist frequency of all spectra in the output SDS product shall be the same. | |
| **FXFORM-OUT-040** | The data type of the flux column of the output spectra shall be either complex or double. | |
| **FXFORM-OUT-041** | If the "ftType" keyword argument is set to "ds" then the flux column shall be complex. | |
| **FXFORM-OUT-042** | If the "ftType" keyword argument is set to "ss" then the flux column shall be double. | |
| **FXFORM-OUT-050** | The spectral sampling interval of the spectra in the output SDS product shall be fixed for a given type of observation. | |
| **FXFORM-OUT-051** | If the building block indicates a low resolution observation, the spectral sampling rate produced by this module shall be equal to $0.25cm^{-1}$. | |
| **FXFORM-OUT-052** | If the building block indicates a medium resolution observation, the spectral sampling rate produced by this module shall be equal to $0.05cm^{-1}$. | |
| **FXFORM-OUT-053** | If the building block indicates a high resolution observation, the spectral sampling rate produced by this module shall be equal to $0.01cm^{-1}$. | |

### 7.9.5 Functional Requirements

| | | |
|---|---|---|
| **FXFORM-FUN-010** | If the "ftType" input argument indicates a doublesided transform is to be computed, only the doublesided portion of the input interferograms shall be used to compute the output spectra. | |
| **FXFORM-FUN-011** | If the "ftType" input argument indicates a singlesided transform is to be computed, only the singlesided portion of the input interferograms shall be used to compute the output spectra. | |
| **FXFORM-FUN-020** | The spectra computed by this module shall be normalized to unit wavenumber. | |

## 7.10 Spectral Response Correction (Spectrometer)

### 7.10.1 Module Owner

Blue Sky Spectroscopy (Trevor Fulton).

### 7.10.2 Module Description

The purpose of this module is to correct each input spectrum for the spectral response of the SPIRE spectrometer.

### 7.10.3 Data

#### *7.10.3.1 Input Data*

| | | |
|---|---|---|
| **SPECRESP-INP-010** | Input science data has been observed using an astronomical observation template AOT. | |
| **SPECRESP-INP-020** | Input science data shall be in the form of a Spectrometer Detector Spectrum (SDS) product. | SDS |
| **SPECRESP-INP-021** | The input SDS product shall have its "type" metadata parameter set to "SDS". | |
| **SPECRESP-INP-022** | The input SDS product shall have its "numScans" metadata parameter set number of scans it contains. | |
| **SPECRESP-INP-023** | The input SDS product shall have its "commandedResolution" metadata parameter set to "HR", "MR", or "LR". | |
| **SPECRESP-INP-024** | Detectors denoted as dead or noisy have had their respective sample masks set for all of their samples. | |
| **SPECRESP-INP-025** | The input SDS product shall have its "actualResolution" metadata parameter set to the resolution of its spectra. This quantity shall be expressed in units of wavenumbers ($cm^{-1}$). | |
| **SPECRESP-INP-030** | The spectral resolution and sample wavenumbers in the input science data shall be the same for all spectra. | |

#### *7.10.3.2 Input Calibration Products*

| | | |
|---|---|---|
| **SPECRESP-CAL-010** | A calibration product that identifies illuminated detectors and unconnected channels shall be provided. | |
| **SPECRESP-CAL-020** | A calibration product that contains the relative spectral response function RSRF for each spectrometer detector shall be provided. | |
| **SPECRESP-CAL-021** | The RSRF calibration product shall contain, for each spectrometer detector, an RSRF for each of the LR, MR, and HR observation types. | |

#### *7.10.3.3 Output Data*

| | | |
|---|---|---|
| **SPECRESP-OUT-020** | The output science product shall be in the form of a Spectrometer Detector Spectrum (SDS) product. | SDS |

| | | |
|---|---|---|
| **SPECRESP-OUT-021** | The output SDS product shall have its "type" metadata parameter set "SDS". | |
| **SPECRESP-OUT-022** | The output SDS product shall have its "numScans" metadata parameter set to the number of scans it contains. This value shall be the same as that for the input SDS product. | |
| **SPECRESP-OUT-023** | The output SDS product shall have its "commandedResolution" metadata parameter set to the same value as that for the input SDS product. | |
| **SPECRESP-OUT-024** | Detectors denoted as dead or noisy in the input SDS product will likewise be denoted as such in the output SDS product and will have their respective sample masks set for all of their samples. | |
| **SPECRESP-OUT-025** | The output SDS product shall have its "actualResolution" metadata parameter set to the resolution of its spectra. This quantity shall be expressed in units of wavenumbers ($cm^{-1}$) and shall be the same as that set in the input SDS product. | |
| **SPECRESP-OUT-030** | The wavenumber column for a given detector in the output SDS product shall be identical to any of the wavenumber columns for that detector in the input SDS product. | |

### 7.10.4 Functional Requirements

| | | |
|---|---|---|
| **SPECRESP-FUN-010** | The module shall remove the RSRF from each spectrum for each spectrometer detector in the input product. | |
| **SPECRESP-FUN-020** | Each spectrum for each spectrometer detector in the input product will have its spectral intensity divided by the RSRF for that spectrometer detector at the commanded resolution as defined in the input calibration product. | |

## 7.11 Spectral Flux Conversion

### 7.11.1 Module Owner

Blue Sky Spectroscopy (Trevor Fulton).

### 7.11.2 Module Description

The purpose of this module is to correct each input spectrum for the spectral response of the SPIRE spectrometer.

### 7.11.3 Data

#### *7.11.3.1 Input Data*

| | | |
|---|---|---|
| **SPECFLUX-INP-010** | Input science data has been observed using an astronomical observation template AOT. | |
| **SPECFLUX-INP-020** | Input science data shall be in the form of a Spectrometer Detector Spectrum (SDS) product. | SDS |
| **SPECFLUX-INP-021** | The input SDS product shall have its "type" metadata parameter set to "SDS". | |

| | | |
|---|---|---|
| **SPECFLUX-INP-022** | The input SDS product shall have its "numScans" metadata parameter set number of scans it contains. | |
| **SPECFLUX-INP-023** | The input SDS product shall have its "commandedResolution" metadata parameter set to "HR", "MR", or "LR". | |
| **SPECFLUX-INP-024** | Detectors denoted as dead or noisy have had their respective sample masks set for all of their samples. | |
| **SPECFLUX-INP-025** | The input SDS product shall have its "actualResolution" metadata parameter set to the resolution of its spectra. This quantity shall be expressed in units of wavenumbers ($cm^{-1}$). | |
| **SPECFLUX-INP-030** | The spectral resolution and sample wavenumbers in the input science data shall be the same for all spectra. | |

#### *7.11.3.2 Input Calibration Products*

| | | |
|---|---|---|
| **SPECFLUX-CAL-010** | A calibration product that identifies illuminated detectors and unconnected channels shall be provided. | |
| **SPECFLUX-CAL-020** | A calibration product that contains the the scaling factors required to convert from voltages to units of optical power for each spectrometer detector shall be provided. | |
| **SPECFLUX-CAL-021** | The calibration product shall contain, for each spectrometer detector, and for each spectral bin, a scaling factor for each of the LR, MR, and HR observation types. | |

#### *7.11.3.3 Output Data*

| | | |
|---|---|---|
| **SPECFLUX-OUT-020** | The output science product shall be in the form of a Spectrometer Detector Spectrum (SDS) product. | SDS |
| **SPECFLUX-OUT-021** | The output SDS product shall have its "type" metadata parameter set "SDS". | |
| **SPECFLUX-OUT-022** | The output SDS product shall have its "numScans" metadata parameter set to the number of scans it contains. This value shall be the same as that for the input SDS product. | |
| **SPECFLUX-OUT-023** | The output SDS product shall have its "commandedResolution" metadata parameter set to the same value as that for the input SDS product. | |
| **SPECFLUX-OUT-024** | Detectors denoted as dead or noisy in the input SDS product will likewise be denoted as such in the output SDS product and will have their respective sample masks set for all of their samples. | |
| **SPECFLUX-OUT-025** | The output SDS product shall have its "actualResolution" metadata parameter set to the resolution of its spectra. This quantity shall be expressed in units of wavenumbers ($cm^{-1}$) and shall be the same as that set in the input SDS product. | |
| **SPECFLUX-OUT-030** | The wavenumber column for a given detector in the output SDS product shall be identical to any of the wavenumber columns for that detector in the input SDS product. | |

### 7.11.4 Functional Requirements

| | | |
|---|---|---|
| **SPECFLUX-FUN-010** | The module shall convert the flux quantities in spectra of the input SDS product from voltage quantities to optical power quantities. | |

| | | |
|---|---|---|
| **SPECFLUX-FUN-020** | Each spectral sample in each spectrum for each spectrometer detector in the input product will have its spectral intensity divided by the scaling factor for that spectrometer detector at the commanded resolution as defined in the input calibration product. | |

## 7.12 Remove Optical Crosstalk (Spectrometer)

## 7.13 Spectral Averaging

### 7.13.1 Module Owner

Blue Sky Spectroscopy (Trevor Fulton).

### 7.13.2 Module Description

The purpose of this module is to create a set of average spectra from a set of input spectra for each spectrometer detector channel for a given observation building block.

### 7.13.3 Data

#### *7.13.3.1 Input Data*

| | | |
|---|---|---|
| **SPECAVG-INP-010** | Input science data has been observed using an astronomical observation template AOT. | |
| **SPECAVG-INP-020** | Input science data shall be in the form of a Spectrometer Detector Spectrum (SDS) product. | SDS |
| **SPECAVG-INP-021** | Input science data shall have its “type” metadata parameter set SDS. | |
| **SPECAVG-INP-022** | Input science data shall have its "numScans" metadata parameter set to a value equivalent to the number of scans it contains. | |
| **SPECAVG-INP-023** | Input science data shall have its "commandedResolution" metadata parameter set to "HR", "MR", or "LR". | |
| **SPECAVG-INP-024** | Detectors denoted as dead or noisy have had their respective sample masks set for all of their samples. | |
| **SPECAVG-INP-025** | Input science data shall have its "actualResolution" metadata parameter set to the resolution of its spectra. This quantity shall be expressed in units of wavenumbers ($cm^{-1}$). | |
| **SPECAVG-INP-030** | The flux samples for each spectrum in the input SDS product shall be real. | |
| **SPECAVG-INP-031** | The wavenumber columns for each scan for a given detector in the input SDS product shall be identical. | |

#### *7.13.3.2 Input Calibration Data*

| | | |
|---|---|---|
| **SPECAVG-CAL-010** | A calibration product that identifies illuminated detectors and unconnected channels shall be provided. | |

#### *7.13.3.3 Output Data*

| | | |
|---|---|---|
| **SPECAVG-OUT-020** | The output science product shall be in the form of a Spectrometer Detector Spectrum (SDS) product. | SDS |
| **SPECAVG-OUT-021** | The output SDS product shall have its "type" metadata parameter set "SDS". | |
| **SPECAVG-OUT-022** | The output SDS product shall have its "numScans" metadata parameter set to 1. | |
| **SPECAVG-OUT-023** | The output SDS product shall have its "commandedResolution" metadata parameter set to the same value as that for the input SDS product. | |
| **SPECAVG-OUT-024** | Detectors denoted as dead or noisy in the input SDS product will likewise be denoted as such in the output SDS product and will have their respective sample masks set for all of their samples. | |
| **SPECAVG-OUT-025** | The output SDS product shall have its "actualResolution" metadata parameter set to the resolution of its spectra. This quantity shall be expressed in units of wavenumbers ($cm^{-1}$). | |
| **SPECAVG-OUT-030** | The quantities in the flux column of the output SDS product shall be real (type Double). | |
| **SPECAVG-OUT-031** | The wavenumber column for a given detector in the output SDS product shall be identical to any of the wavenumber columns for that detector in the input SDS product. | |

### 7.13.4 Functional Requirements

| | | |
|---|---|---|
| **SPECAVG-FUN-010** | For each detector in the input product, the module shall compute the mean value of the spectral flux samples at each wavenumber sample across all scans. | |
| **SPECAVG-FUN-011** | For each detector in the input product, the module shall compute the standard deviation of the spectral flux samples at each wavenumber sample across all scans. | |
| **SPECAVG-FUN-020** | The module shall be able to filter outliers on a spectral sample by spectral sample basis before computing the average and standard deviation. | |
| **SPECAVG-FUN-021** | Outliers shall be defined as those samples that fall outside the range of plus or minus three scaled median absolute deviations from the median of all of the samples for a given detector in a given wavenumber bin. | |

## 7.14 Spatial Regridding

### 7.14.1 Module Owner

Blue Sky Spectroscopy (Ralph Boland).

### 7.14.2 Module Description

The purpose of this module is to create a spectral cube that is equidistantly gridded in all three dimensions, i.e. the two spatial and the one spectral dimension for data from a set of observation building blocks derived from a jiggle or raster observation. The module can also be applied to sparsely sampled data. However, in this case, the garbage-in-garbage-out rule may apply.

### 7.14.3 Data

#### *7.14.3.1 Input Data*

| | | |
|---|---|---|
| **SPECCUB-INP-010** | Input science data have been observed using an astronomical observation template AOT. | |
| **SPECCUB-INP-020** | Input science data shall be in the form of an array of averaged Spectrometer Detector Spectrum (SDS) products. | SDS |
| **SPECCUB-INP-021** | The products in the array of input science data shall have their "type" metadata parameter set to "SDS". | |
| **SPECCUB-INP-022** | The products in the array of input science data shall have their "commandedResolution" metadata parameter set to "HR", "MR", or "LR". | |
| **SPECCUB-INP-023** | The products in the array of input science data shall have their "actualResolution" metadata parameter set to the resolution of its spectra. This quantity shall be expressed in units of wavenumbers ($cm^{-1}$). | |
| **SPECCUB-INP-024** | The products in the array of input science data shall have their "instrument" metadata parameter set to "SPIRE". | |
| **SPECCUB-INP-025** | The world coordinates of the sky position is provided for each detector spectrum in the input science data. | |
| **SPECCUB-INP-026** | The flux column in the array of input science data for all detectors shall have the same unit. | |
| **SPECCUB-INP-030** | The flux samples for each spectrum in each one of the input SDS products shall be real. | |
| **SPECCUB-INP-031** | The error samples for each spectrum in each one of the input SDS products shall be of the same data type as the flux samples. | |
| **SPECCUB-INP-032** | The error samples for each spectrum in each one of the input SDS products reflect the uncertainty of the flux samples. | TBC whether zeros are OK. |
| **SPECCUB-INP-033** | The SDS products in the array of input science data shall contain identical wavescales. | |

#### *7.14.3.2 Input Calibration Data*

| | | |
|---|---|---|
| **SPECCUB -CAL-010** | A calibration product that details the beam profiles of the spectrometer detectors at various wavelengths across the optical passbands shall be provided. | |

#### *7.14.3.3 Output Data*

| | | |
|---|---|---|
| **SPECCUB-OUT-010** | The science output data shall be in the form of two Spectral Cube products, one for the SLW array and one for the SSW array. | SpectralCube |
| **SPECCUB-OUT-011** | The output cube products shall have their "type" metadata parameter set to "SpectralCube". | |
| **SPECCUB-OUT-012** | The output cube products shall have their "unit" metadata parameter set to the flux unit specified in the input science data SDS products. | |
| **SPECCUB-OUT-013** | The output cube products shall have their "instrument" metadata parameter set to "SPIRE". | |
| **SPECCUB-OUT-014** | The output cube products shall have their "description" metadata parameters set to "SLW" or "SSW". | |
| **SPECCUB-OUT-020** | The wavenumber column for a given detector in each output cube product shall be identical to any of the wavenumber columns for | |

| | | |
|---|---|---|
| | that detector in the input SDS. | |
| **SPECCUB-OUT-021** | The spatial coordinates in each output cube product will be provided in equidistant arrays of coordinates RA and DEC. | |

### 7.14.4 Functional Requirements

| | | |
|---|---|---|
| **SPECCUB-FUN-010** | The module will process one two-dimensional slice of a spatial image at a time. | Testable |
| **SPECCUB-FUN-011** | The module will take the varying beam profile for a given frequency into account. | Testable |
| **SPECCUB-FUN-012** | The module will perform a two-dimensional convolution. | Not testable |
| **SPECCUB-FUN-013** | The module will conserve the flux density of the input spectra. | Testable |

# 8. QUALITY CONTROL

The Herschel ICCs are responsible for providing software for the processing of observational data into standard products for distribution to observers. This software is run as a pipeline within the ESA SPG pipeline environment and consists of a series of modules called in sequence to progressively process the data towards a more scientifically useful product. AD01 describes the SPIRE pipline(s) in detail.

This module is used to gather together quality control information from the various modules in the pipeline. The module is intended for use on the level 0.5, 1, and 2 products produced by the pipeline, although the module has been designed so that it may be used on any Product, including products produced at intermediate stages between various levels.

## 8.1 Module Owner

Imperial College London (George J. Bendo responsible, Davide Rizzo contributing)

## 8.2 Others Contributing

## List of Applicable Documents

AD01 SPIRE Pipeline Description (SPIRE-RAL-DOC-002437)
AD02 SPG Pipeline ICD

## 8.3 List of Reference Documents

RD-01 SPIRE Quality Control Requirements

## 8.4 Module Description

The purpose of this module is to gather together quality control metadata from the various pipeline modules and to produce a quality product.

## 8.5 Assumptions

### Input data

The following assumptions are made

- Input data is in the form of a Product.
- When present, quality control information is included in the Product as metadata.
- In the input data, the names, variable types, and accepted ranges for the quality control metadata match what other module developers have specified.

## 8.6 Requirements

Module requirements are taken from

- ICC Use Cases
- AD01: Pipeline description document
- ICC Discussions

## Functional Requirements

| | |
|---|---|
| **QUAL-FUN-010** | The module shall operate on any Product |
| **QUAL-FUN-020** | The module will report a complete set of quality control parameters |
| **QUAL-FUN-030** | The module will report null values when any specific metadata keywords are not present in the input Product |
| **QUAL-FUN-040** | The module shall produce Boolean parameters for each quality control parameter to show whether each parameter falls within an acceptable range (with true indicating that the parameter is unacceptable) |
| **QUAL-FUN-050** | The module will place all output within a QualityContext |

# 9. PCAL

This module is one of the pre-processing steps used to process PCAL flashes to provide a measurement of detector relative sensitivity during an observation. It is expected that this pre-processing will be carried out on each Operational Day's data prior to running the standard pipeline. The PCAL output products (one per Building Block) may be used for trend analysis. The SPG pipeline will have the option to use the information stored in the PCAL output product to account for changes in responsivity of the bolometers with time during an observation; at present this is not the baseline. The PCAL output may also be used for nonlinearity measurements, when a bright astronomical source is imaged on the detectors during the flashes. The PCAL processing may occur relatively early in the processing (e.g. on Level 0.5 data in units of Volts), or it could potentially be run after non-linearity and flux conversion have been performed.

## 9.1 Module Owner

NHSC (Arnold Schwartz: implementation; David Shupe & Bernhard Schulz, science requirements).

## 9.2 Others Contributing

TBD.

## 9.3 Module Description

The purpose of this module is to process a single PCAL flash building block and to output tables of processed values, to be used for trend analysis, and if necessary by subsequent pipeline processing steps. The module will measure mean PCAL signal levels and will characterize the detector time constants for the PCAL flash turning on and off.

## 9.4 Data

### 9.4.1 Input data

The PCAL module requires two input data products. The first input data product will be in the form of a Detector Timeline (class herschel.spire.ia.dataset.DetectorTimeline) taken from a single Building Block, photometer or spectrometer as appropriate, with additional processing if necessary. These data will nominally have been processed by first level deglitching and, if necessary, electrical crosstalk removal. Alternatively, the input may have been run through the bolometer model, or may have been run through nonlinearity correction and flux conversion. Accordingly, the input units of the signal table in the product can be Volts, Watts, or Janskys—the PCAL module will propagate the input units to the output product.

The second input data product is the SCUT timeline from the same Building Block. The SCUT timeline contains the PCAL current and voltage timelines.

### 9.4.2 Input Calibration Products

The input calibration product will contain the following parameters:

- A table of levels, including for each envisaged PCAL sequence:
  - Number of flashes in the Building Block
  - Commanded electrical PCAL voltage, upper limit tolerance, lower limit tolerance
  - Commanded electrical PCAL current, upper limit tolerance, lower limit tolerance
- For each detector, expected flash-on and flash-off nominal time constants and their upper limits (4 values per detector)

The input calibration product will be in the form of either a PhotPcalPar or SpecPcalPar (TBC) product. It will be derived using a script run by a calibration scientist (makePcalPar.py).
*PFM4 AOT and ILT tests employed a total of 8 PCAL voltage-current-number combinations for PHOT and 6 such combinations for SPEC. These combinations will be placed in the initial table of levels.*
*If an "uplink" table of commanded values becomes available, then the information on number of flashes, and flash levels, may be taken from that input instead.*

### 9.4.3 Output Data

The output of the PCAL module will be in the form of either the PhotPcal or SpecPcal product. One product will be produced for each BBID. The products will be based on the PixelCalibTable class. The class definitions reside in herschel.spire.ia.dataset.

## 9.5 Functional Requirements

**PCAL-FUN-010** The module shall process data from a single PCAL Flash Building Block.

**PCAL-FUN-020** The module shall be able to generate data from both SPIRE photometer and spectrometer detectors.

**PCAL-FUN-030** Using the detector signals of PCAL flashes that make up one PCAL building block, the module shall determine a set of parameters that are relevant for calibration and trend analysis.
*This pattern of flashes is assumed to be a signal alternating between two levels for a given time at a given frequency.*

**PCAL-FUN-040** The module shall be able to process data from different PCAL flash levels, frequencies and number of flashes. The module shall determine the timing of PCAL flashes from the SCU timeline in the Building Block. The processing shall assume that only one PCAL "flash-on" level is used within a single PCAL Building Block.
*All AOT and ILT tests in PFM3, PFM4 and PFM5 used a single "flash-on" current level within a single Building Block. ILT tests exercised two PCAL "on" current levels, but these took place in separate Building Blocks.*

**PCAL-FUN-050** The module shall calculate the stabilization time constant (i.e. the time constant of a fitted exponential to the signal after each change of PCAL- illumination), assuming that the PCAL flash is a step function with an exponential drop or rise between steps. The module shall report anomalous variations in the signal rise or decay time constant with test limits specified as input parameters in an input calibration product.
*Specifically this should identify contamination by helium build up on the bolometers, which would result in a detector time constant longer than that of PCAL turning on or off.*

**PCAL-FUN-060** The module shall produce a PCAL Output Table product, which shall include the calculated quantities.

**PCAL-FUN-070** The calculated quantities are:
number of flashes found
average power input to PCAL, in Watts
For each detector:
- average & rms base signal (no PCAL illumination)
- average & rms signal difference
- average & rms time constant on for "flash-on"
- average & rms time constant for "flash-off"
- flag for successful fit of "flash-on" time constant
- flag for successful fit of "flash-off" time constant.

The units of the base signal and signal difference quantities shall be the same as the units in the input detector timeline. The units of the time constant quantities shall be in milliseconds.

**PCAL-FUN-080** The module shall compare the calculated quantities with nominal values and tolerances given by an input calibration product.

**PCAL-FUN-090** The input calibration product will contain the following parameters:

- A table of levels including:
  - Number of flashes expected in each Building Block (1 value)
  - Expected electrical PCAL voltage, upper limit tolerance, lower limit tolerance (3 values)
  - Expected electrical PCAL current, upper limit tolerance, lower limit tolerance (3 values)
- For each detector, a table of expected flash-on and flash-off nominal values and upper limits (4 values per detector)

**PCAL-FUN-100** The module shall flag out-of-range values in the PCAL Output Table product. Specifically, the output product shall report PCAL current or voltage levels that are outside the upper and lower limits, and for each detector, flash-on or flash-off time constants that exceed the upper limits.

# 10. POINT SOURCE EXTRACTION

This module is one of the final processing steps used to process map data produced by the instrument to provide a list of point sources. As both SPIRE and PACS are able to produce maps from their observations and these should be as far as possible instrument independent, it is expected that this software will be able to deal with data from both instruments although some instrument specific processing may be required.

## 10.1 Module Owner

RAL (Huw Morris)

## 10.2 Others Contributing

Sussex University (Rich Savage developed the original algorithm, now being maintained by TBC)

## 10.3 Module Description

The purpose of this module is to process a map of an area of sky to provide a list of point sources identified within the map.

## 10.4 Assumptions

### 10.4.1 Input data

The following assumptions are made

- Input data is in the form of a map.
- Input data may contain a pixel mask. (dead pixels, etc)
- Input data is in the form of calibrated data samples with instrumental effects removed.
- Some measure of coverage is provided
- Input data has good astrometric calibration (including header information to convert between sky co-ordinates and map co-ordinates)

- Input data has good flux calibration (including a statement of the units)
- Input data has good quality flags
- Input data has a measure of the statistical weight (i.e. inverse variance) of each datum.
- Input data has a measure of covariance, if non-negligible
- Input data has well-quantified noise characteristics (for example, ideally we want minimal noise correlation between data samples; where there are correlations, we want then quantified)
- Input data includes the point spread function associated with the observation in question
- Input data can optionally include a Source List Product
- Map data includes the angular size of the (square) pixels

### 10.4.2 Output Data

The module will produce a Source List Product as its output.

## 10.5 Functional Requirements

**SRCEXT-FUN-010** The module shall operate on data from a single map product

**SRCEXT-FUN-020** The module shall be able to deal with both SPIRE and PACS mapping data products

| | | |
|---|---|---|
| **SRCEXT-FUN-030** | The module shall take as input an Image Product. The capability to process a Timeline Product is desirable<br>*SPIRE maps are produced as Image Products, PACS maps are produced as TBD products* | |
| **SRCEXT-FUN-040** | The module shall be capable of identifying, and extracting photometric parameters for, point sources from a given band in the input map, conforming to the source detection threshold and other input parameters provided | |
| **SRCEXT-FUN-050** | The module shall be capable of extracting photometric parameters for sources in a list provided as input (in a Source List Product). | |

**SRCEXT-FUN-060** The module shall take as parameters the following:

- a source detection threshold (in terms of S/N or Beysian evidence)
- the band in which the detection is performed
- the band for which photometric parameters will be extracted
- sub-region of the map to be considered for processing around each pixel
- sub-region of the map to be processed, specified in RA, Dec

*These parameters may be provided by the pipeline or the user operating interactively*

| | | |
|---|---|---|
| **SRCEXT-FUN-070** | The module shall take the point spread function as an optional parameter. Where no PSF is specified, a default PSF shall be used. Each instrument will have a different default. | |
| **SRCEXT-FUN-080** | The module will be designed such that the exact algorithm for source extraction is pluggable.<br>*Other implementations, such as DAOPHOT, will be tested against Sussextractor.* | |

| | | |
|---|---|---|
| **SRCEXT-FUN-090** | The module shall produce a Source List Product as output<br>*This product shall contain at least the following information:*<br>• *Source position, including errors*<br>• *Estimated source flux, including error for detected band*<br>• *Estimated Source size, including errors (TBC)*<br>• *Background parameters (TBD)*<br>• *Source quality*<br>• *TBD* | |

# 11. ADDITIONAL MODULES

This section contains additional modules which are not part of the main pipeline at this point. Many of these modules are part of the 'bolometer physical model' approach to bolometer processing rather than the 'empirical model' method being used for the current version of the pipeline. Others are related to Level 3 products which allow for additional astrophysical data processing but which are not intended for the automatic SPG version of the pipeline. These modules may be implemented at a later date.

## 11.1 Conversion to a Different Spectral Index

## 11.2 Correct for Non-Linearity

## 11.3 Calculate and Remove Background Power/Signal

### 11.3.1 Module Owner

Gabriele Mainetti, Dipartimento di Astronomia, Università di Padova

### 11.3.2 Others Contributing

Mattia Vaccari, Dipartimento di Astronomia, Università di Padova

### 11.3.3 Module Description

The purpose of this module is to compute and (TBC) remove signal contributions due to the telescope background emission in SPIRE photometer observations carried out in scan map mode. Following AD01 and AD02, the computation relies on calibration measurements of background emission in "dark" conditions acquired at the same telescope and FPU temperatures. This document details requirements characterizing this specific module and the data products required and produced by the module. Module requirements are taken from AD01, AD02 and AD03. Data products are described in detail in AD05.

### 11.3.4 Data

#### *11.3.4.1 Input Data Products*

Photometer Detector Timeline (AD05)

Telescope Temperature Timeline (TBD at Herschel-wide level under PACS responsability)

#### *11.3.4.2 Input Calibration Products*

"Dark" Background Emission Calibration Data Products at a number of relevant telescope and FPU temperatures (TBD by SPIRE CALT)

#### *11.3.4.3 Output Data Products*

Background subtracted Photometer Detector Timeline.

### 11.3.5 Functional Requirements

**PHOTBKGD-FUN-010** The module shall receive as input, operate on and produce as output, data in the form of a photometer detector timeline produced in scan map observations.

**PHOTBKGD-FUN-020** The module shall identify input calibration data products acquired under similar conditions (telescope and FPU temperatures, detector bias voltage and frequency) and, where necessary, take into account small differences in these parameters.

**PHOTBKGD-FUN-030** The module shall compute background emission in "dark" conditions following AD01, namely Section (5.4.1) and Equations (25) and (29) (for the empirical pipeline) and Section (6.5) and Equations (59) and (60) (for the model-based pipeline).

**PHOTBKGD-FUN-040** The module shall allow the toggling of the actual subtraction of computed background emission according to user's input.

## 11.4 Calculate Temperature and Conductance

### 11.4.1 Module Owner

CEA Saclay (René Gastaud responsible).

### 11.4.2 Others Contributing

Currently none.

### 11.4.3 Module Description

The purpose of this module is to compute the Bolometer Heat Sink Temperature. This will be used by the subsequent pipeline processing step: calculate absorbed power. The temperature is computed using the resistance of each thermistor, and is then filtered to get rid of noise.

### 11.4.4 Data

#### *11.4.4.1 Input data*

There are two inputs: the Detector Timeline and the housekeeping timeline NHKT.

- DT: can be a Photometer or a Spectrometer timeline. It must contain the resistance of the thermistors.
- NHKT: contains bias voltages.

#### *11.4.4.2 Input calibration files*

This module also requires the calibration product Bolometer Parameters, DPCR-13.

#### *11.4.4.3 Output data*

There is only one output data product. This will contain the Bolometer Heat Sink Temperature. For each bolometer it gives the heat sink temperature at a given time. This is described in SPIRE Data Products Specification as BTT.

#### *11.4.4.4 Functional Requirements*

| | |
|---|---|
| **CalculateTemperature-FUN-010** | The module shall process data from a single DT |
| **CalculateTemperature - FUN-020** | The module shall compute the heat sink temperature using the best method (SPIRE-UCF-DOC-002890, The SPIRE Analogue Signal Chain and Photometer Detector Data Processing Pipeline Matt Griffin). |
| **CalculateTemperature - FUN-030** | The calculated quantities are:<br>The heat sink temperature for each subarray. It will be stored in a BTT. |
| **CalculateTemperature - FUN-40** | The user can select the time filter. |

## 11.5 Calculate Absorbed Power

### 11.5.1 Module Owner

CEA Saclay (René Gastaud responsible).

### 11.5.2 Others Contributing

Currently none.

### 11.5.3 Module Description

The purpose of this module is to compute the power absorbed by each active bolometer. It uses a model of the physics of the bolometer, described in SPIRE-UCF-DOC-002890. The SPIRE Analogue Signal Chain and Photometer Detector Data Processing Pipeline, Matt Griffin.

### 11.5.4 Data

#### *11.5.4.1 INPUT DATA*

There are three inputs: the Detector Timeline DT, the housekeeping timeline NHKT, the Bolometer Heat Sink Temperature BTT.

- DT: this can be a Photometer or a Spectrometer timeline. It must contain the resistance and the voltage of each active bolometer.
- NHKT: this contains bias voltage timelines.
- BTT: this gives the heat sink temperature at a given time for each bolometer.

#### *11.5.4.2 Input calibration files*

This module also requires the calibration product Bolometer Parameters, DPCR-13.

#### *11.5.4.3 Output data*

There is only one output data product, a detector timeline, which will contain the power instead of the resistance and the voltage for each bolometer.

#### *11.5.4.4 Functional Requirements*

| | |
|---|---|
| **CalculatePower-FUN-010** | The module shall process data from a single DT |
| **CalculatePower -FUN-020** | The module shall compute the absorbed power with the physics model described in SPIRE-UCF-DOC-002890, The SPIRE Analogue Signal Chain and Photometer Detector Data Processing Pipeline, Matt Griffin. |
| **CalculateTemperature -FUN-030** | The calculated quantities are: the absorbed power for each active bolometer. It will be stored in a DT. |
| **CalculateTemperature -FUN-40** | If the power is negative, an error is set. |

# 12. CHANNEL FRINGE CORRECTION (NOT A BASELINE DELIVERY)

## 12.1 Module Owner

Blue Sky Spectroscopy (Trevor Fulton).

## 12.2 Others Contributing

None

## 12.3 Module Description

The purpose of this module is to remove from the set of input interferograms the contributions that are due to the instrument channel fringes.

## 12.4 Data

### 12.4.1 Input Data

Input science data product is a set of interferograms whose samples are regularly spaced in terms of OPD.

Input interferograms for a given spectrometer detector channel contain the same number of samples and are sampled on a common set of positions.

Input interferogram data has been converted from raw sample counts to at least engineering units.

Input data has good quality flags.

### 12.4.2 Output Data

## 12.5 Functional Requirements

| | | |
|---|---|---|
| **CHANFRNG-FUN-010** | The module shall operate on data from a single spectrometer observation building block. | |
| **CHANFRNG-FUN-020** | The module shall be able to operate on both single-sided and double-sided interferograms. | |
| **CHANFRNG-FUN-030** | The module shall remove the position-dependant baseline from the input interferograms to within 1% of the mean value of the input interferogram at the position of zero path difference. | |