SUBJECT:	**FTS Pipeline Scientific Validation, Phase 3 Report**		
PREPARED BY:	Ed Polehampton, Peter Davis-Imhof, Jean-Paul Baluteau		
DOCUMENT No:	SPIRE-RAL-DOC-003271		
ISSUE:	1.0	Date:	22nd October 2009
APPROVED BY:		Date:	

Distribution

Jean-Paul Baluteau	LAM
Peter Davis-Imhof	Blue Sky Spectroscopy
Ed Polehampton	RAL
Peter Ade	Cardiff University
Trevor Fulton	Blue Sky Spectroscopy
Nanyao Lu	IPAC
David Naylor	University of Lethbridge
Giorgio Savini	University College London
Christian Surace	LAM
Bruce Swinyard	RAL
Dominique Benielli	LAM
Scott Jones	University of Lethbridge
Sarah Leeks	RAL
Chris Pearson	RAL
Tanya Lim	RAL
Matt Griffin	Cardiff University
Michael Pohlen	Cardiff University
Pasquale Panuzzo	CEA

Change Record

ISSUE	DATE	Changes
Draft 1.0	18 June 2009	
Draft 1.1	01 August 2009	
Issue 1.0	22 October 2009	

1 INTRODUCTION

1.1 The SPIRE FTS Validation Group

1.1.1 Group Membership

Coordinators:

Ed Polehampton (RAL)
Jean-Paul Baluteau (Marseille)
Peter Davis-Imhof (Blue Sky Spectroscopy)

Members:

Peter Ade (Cardiff)
Trevor Fulton (Blue Sky Spectroscopy)
Nanyao Lu (IPAC)
David Naylor (Lethbridge)
Giorgio Savini (Cardiff)
Bruce Swinyard (RAL)
Christian Surace (Marseille)
Dominique Benielli (Marseille)
Scott Jones (Lethbridge)

Cross-members (coordinating across all 4 groups):

Sarah Leeks (RAL)
Chris Pearson (RAL)

1.1.2 Objectives

The Objectives of the validation group are:

1. To ensure the pipeline conforms to the top-level documentation in terms of their overall architecture and detailed implementation.
2. To ensure that the developer documentation for individual modules conforms to the top-level documentation in terms of requirements and algorithms.
3. To verify that testing carried out at the developer module level is adequate and documented
4. To test the pipeline to validate the correct operation of individual modules and end-to-end systems.
5. To identify and initiate correction of errors or omissions in the pipeline documentation.
6. To identify and report errors in the module implementation and operation.
7. To document all results from the test phases.

The software test of pipeline modules aims to check:

- Consistency with the (already reviewed) top-level documents and module requirements
- Consistency with calibration file definitions (as described in the Pipeline Description Document)
- Correctness and clarity of implementation (i.e. algorithms used are correct and method clear)
- Commonality in use of symbols and terminology (i.e. inputs/outputs to each module use consistent terminology, algorithms use consistent symbols)
- Status of module-level testing (i.e. testing that has been carried out so far)

1.2 Structure of this Document

This document contains a review of the Spire Spectrometer pipeline as a whole. The investigation consists of three main sections: the order of modules in the pipeline, the calibration products, and the user guide (including consistency of naming conventions). This document contains a summary of the discussions on each of these issues, a list of recommendations and a link to Spire SxR numbers where the issues have already been raised.

1.3 Documents

1.3.1 Applicable Documents

1.3.2 Reference Documents

User Guide	SPIRE Pipeline User Guide, version 0.02, 18 September 2008
Chris' Document	SPIRE Pipeline Description (SPIRE-RAL-DOC-002437) Issue 2.1, 8 May 2009
Trevor's Document	SPIRE Spectrometer Pipeline Description (SPIRE-BSS-DOC-002966) Issue 1.3.1, 28 June 2008
Module Requirements	SPIRE Data Processing Pipeline Module Requirements (SPIRE-ICS-DOC-002998) Draft 2.2, June 2009

2 THE ORDER OF MODULES IN THE PIPELINE

2.1 Basis for discussions

Discussions of the order of modules in the pipeline were conducted on the following basis:

- All of the blocks needed exist already in the pipeline - we need to be flexible in any recommendations about changes because we may not know the optimum order until after launch
- We will discuss a "robust" order that will not cause problems for the standard pipeline. We can optimise later using interactive analysis (these are two different goals: to provide something that works almost all of the time in a standard way, and then to squeeze out all of the possible information from the data)
- The non-linearity and temperature drift corrections are tied together and should be put together (even though in principle they could be separated)
- We need to take account of which effects will actually be most important (some may not actually have a big effect on most data). However, the non-linearity may not look important but is actually critical
- We need to be careful of doing corrections twice to make sure that this does not introduce errors
- The order of modules in the time domain is the most critical to get correct

As background to the discussions, two reports were produced by Giorgio and Jean-Paul with recommendations. These are given in Appendix A and B in their original form. These two reports were discussed and the following sections give our consolidated recommendations.

2.2 Time Domain Modules

2.2.1 Clipping Module

For bright sources, the bright source mode should reduce the chance of clipping. There shouldn't be many sources that fit into this category. The current proposed bright source mode gives a reduction in gain by a factor of 3. We made the following recommendations, but **note that the issue should be re-visited in the light of real PV phase data**, once we know for sure what clipping is actually present in real observations and if the steps to avoid it actually work.

The clipping correction should not be placed in the standard SPG (Standard Product Generation) pipeline, but instead be available for Interactive Analysis. It should not be left commented out in the SPG pipeline script because we do not know the best location for it yet (and it might need to be run twice).

If an observation shows clipping at ZPD, or a clipped glitch, **the entire observation should only be processed to the interferogram stage, and then flagged as needing attention**. This approach is preferred over processing the observation until the end, as it will give a clear signal to the Astronomer that they shouldn't trust the data and need to reprocess it.

If an observation shows clipping at high OPD (but not at ZPD) in some detectors, the observation should be processed as normal but with reduced spectral resolution in the affected detectors (the clipped part should be cut).

We need to figure out where to recommend that people put the clipping correction by analysing the data observed in Commissioning and PV phases. The clipping correction module is complicated and a normal user may need help to apply it. In this case they should be referred to an expert. On the other hand, it may be better to re-observe the source with different detector settings if clipping is bad.

The approach to flagging clipped observations should be verified during PV phase, and also the 8th order polynomial used in the clipping task to reconstruct the data should be verified with in-flight data.

2.2.2 Remaining Modules

If no clipping is done in the SPG, we don't need to run the 1st Level Degliching task twice (as in Jean-Paul's note in Appendix 2).

Electrical Crosstalk needs to be before deglitching, as otherwise it might propagate glitches back onto the detector from which they had already been removed.

Non-Linearity and Bath Temperature Correction are tied together, and Bath Temperature Correction must be put after Non-Linearity. The connection between these two modules is summarised in the following points:

1) The photometer flux conversion module or spectrometer non-linearity correction integrates dQ/dV from a reference voltage V_0 to the measured voltage V to derive an optical load Q (or a linearized voltage in the FTS case) with respect to V_0. This Q may have 2 components: (i) a contribution from the optical flux and (ii) a component that is caused by a bath temperature drift between when you measure V and when you measure V_0. The module does not care about nor know (ii). Suppose V_0 was measured at the time when the thermistor signal is S_0 (which corresponds to some bath temperature T_0). We pass the value of S_0 to the bath temperature drift correction module.

2) In the bath temperature drift correction module, it checks the actual thermistor signal S for the bolometer voltage V under consideration. If S is not equal to S_0, it applies a correction to Q (by either subtracting or adding a small amount that corresponds to the contribution to Q as a result of the bath temperature difference between the thermistor signals S and S_0). The result is a corrected Q (or corrected linearized signal in the case of FTS) that would be the optical load on the detector if the thermistor signal were at S_0. So after the bath temperature correction module, everything refers to this reference thermistor signal S_0 (which of course corresponds to some bath temperature T_0. But T_0 is model dependent and you don't need to know its value).

3) In practice, when we do the nonlinearity calibration, we measure V_0 and write down the thermistor signal S_0 when V_0 was measured. We then use this S_0 in the derivation of the temperature drift correction calibration product. In this way, the two modules are always consistent with each other. Note that no bath temperature values and no bolometer model are actually involved in this process. In other words, all are empirical and based on measurements. Also note that this implementation has been reflected in the latest pre-launch calibration products.

The position of the Time Domain Phase correction does not matter, except that glitches should have been removed first and Bath Temperature correction must be beforehand. This is because the Bath Temperature Drift Correction relies on a comparison with the Thermistor timeline, and this should be done before the phase shift is applied. Therefore, it seems best to do the best correction that we can for non-linearity and then do the Time Domain Phase correction last.

2.2.3 Conclusions

In summary, the module order in the SPG pipeline should be:

1. **Electrical Crosstalk**
2. **Deglitching**
3. **Non-linearity**
4. **Bath Temperature Correction**
5. **Time Domain Phase Correction**

The decision to remove clipping should be revisited once we have real PV phase data and we have made observations using the final configuration of the AOT.

2.3 Spatial and Spectral Domain Modules

2.3.1 Spatial Domain

There may be a problem with the Phase Correction step of the pipeline for low S/N scans. If we do an excellent job subtracting the reference interferogram, we will be left with low S/N in each scan and ZPD may not be obvious. The problem would be if the phase correction module fitted a different phase to each scan due to low S/N (even with a linear fit, this may be a problem). When they are combined later there would be differences between scans.

The solution to this would be to average the interferograms before phase correction. An average interferogram module should not be difficult to implement. The only reason not to average interferograms is to keep forward and reverse scans separate. We should still keep the possibility to process forward & reverse scans independently.

Conclusions from the discussion of the spatial domain:

- Examine PV data to determine if averaging is needed prior to phase correction
- Verify that all necessary calibration products provide separate information for forward/reverse scans (examine PV data before making changes to existing products)

2.3.2 Spectral domain

There may be some technical merit (e.g. less computation) to move the Averaging module to an earlier stage in the spectral domain of the pipeline - i.e. right after the Fourier Transform module.

Some calibration products (such as flux conversion, optical crosstalk, reference interferogram) may depend on BSM position. We should check this carefully in PV data. We also need to check whether there are sufficient PV observations to determine if this is true.

Conclusions from the discussion of the spectral domain:

- Raise SCR to place averaging directly after Fourier Transform
- Examine PV data to determine if calibration products depend on BSM position

2.4 SxRs Raised

The following SxRs have been raised to initiate implementation of the recommendations on the pipeline module order.

SxR number	Title
SPIRE SCR-1420	SciVal-FTS3: change the sequence of the time-domain modules for SOF scripts
SPIRE SCR-1421	SciVal-FTS3: stop processing clipped data
SPIRE SCR-1766	SciVal-FTS3: Place the AverageSpectraTask right after the FourierTransformTask

3 Review of Calibration Products in the Spectrometer Pipeline

This sections details recommendations resulting from a review of the calibration products used in the pipeline. The following table summarises the calibration products from each domain of the pipeline.

Domain	Pipeline Modules	Calibration Products
Temporal	Deglitch	
	Electrical Crosstalk	SCalSpecElecCross
	Non-linearity correction	SCalSpecNonLinCorr
	Clipping	
	Time Domain Phase Correction	SCalSpecLpfPar SCalSpecChanTimeConst
	Temperature drift correction	SCalSpecTempDriftCorr
	Calculate BSM Angles	SCalSpecBsmPos
	SPIRE Pointing Product	SCalSpecDetAngOff
Spatial	Create Interferogram	SCalSpecSmecZpd SCalSpecChanMask SCalSpecChanTimeOff SCalSpecSmecStepFactor
	SCAL correction	SCalSpecInterRef
	Baseline correction	
	2nd Level deglitching	
	Phase correction	SCalSpecNlp SCalSpecBandEdge
	Apodization	
Spectral	Fourier Transform	
	Flux Conversion	SCalSpecFluxConv
	Optical Crosstalk	SCalSpecOptCross
	Averaging	
	Spatial Regridding	SCalSpecBeamProf

3.1 General Comments

The calibration context can be viewed in Hipe using the Context Viewer. We recommend that to improve the presentation of Calibration Products that,

- it would be more convenient if the calibration tables were presented in alphabetical order when viewed in the calibration context
- it should be made more obvious which editions of calibration products refer to which conditions in Hipe

The channels that are included in each calibration product at different stages of the pipeline should meet the following criteria.

Up to the end of the spatial domain, these channels should be included in every product:

Bolometers: SLWA1-SLWE3, SSWA1-SSWG4, SL/SWDP1 and DP2

BDA Thermistors: SL/SWT1 and T2

BDA Resistors: SL/SWR1

The following channels should be excluded:

PTC Thermistors: N/A for the FTS

Other Channels: SSWN1-N6 (Note: these do not appear in the Level-0 SDT anyway)

After the Fourier Transform stage, the following channels should be included:

Bolometers: SLWA1-SLWE3, SSWA1-SSWG4

The following should be excluded:

Dark Bolometers: SL/SWDP1 and DP2

BDA Thermistors: SL/SWT1 and T2

BDA Resistors: SL/SWR1

This will allow processing of thermistor and resistor signals up to the FT module.

3.2 Temporal Domain

SCalSpecElecCross

SCalSpecNonLinCorr

SCalSpecLpfPar

SCalSpecChanTimeConst

SCalSpecTempDriftCorr

SCalSpecBsmPos

SCalSpecDetAngOff

3.3 Spatial Domain

3.3.1 SCalSpecSmecZpd

1) Change the description to "Position of ZPD" since the table dataset is not limited to the Optical Encoder.
2) A discussion is needed as to what the column "lvdt" is used for. Currently this column is filled with real values with units of V. Should this be in ADU – i.e. integers with no units?
3) It may be a good idea to populate the error columns with the estimated precision of the OE. To be conservative these should be set to 15 nm for OE and 200 nm for the lvdt.

3.3.2 SCalSpecChanMask

Calibration product looks complete and correct.

3.3.3 SCalSpecChanTimeOff

Numbers seem to be complete and these were spot-checked for agreement with the ones given in a spreadsheet from CEA.

3.3.4 SCalSpecSmecStepFactor

There is a complete set of values, all of which are set to 4.0.

3.3.5 SCalSpecInterRef

The complete range of OPD is covered at 25μm steps, containing ZPD.

1) The calibration product has many meta-data entries. They should be reviewed as to whether they are applicable at all (e.g. "bbid", but many more)
2) The calibration product depends on biasMode and time. The list of associated products in theSpecInterRefList does not present how the included products depend on biasMode and/or time.
3) The conclusion from discussions within the FTS team was to create one table for each, low, medium, high, and full resolution. This is not yet implemented.
4) Detectors should be listed in alphabetical order
5) It may be useful to set the meta datum "count" of the composite dataset containing the interferograms to the number of contained interferograms (or the number of interferograms used in the average?).

3.3.6 SCalSpecNlp

1) The datasets for the two arrays are given in lower case in contrast to all other calibration products. They should be changed to upper-case SSW and SLW instead.
2) The non-linear phase will quite likely be significantly different for different detectors. The calibration tables should provide the non-linear optical phase per detector.
3) Only the sky phase is needed in the calibration product. If other phases (e.g. SCAL) are needed later, this will be more complicated and should be added at that point.

3.3.7 SCalSpecBandEdge

Looks self-explanatory and correct. Note that SDAG have recommended renaming this product to be "Phase Correction Limits" (as this is all the limits are used for in the pipeline).

3.4 Spectral Domain

3.4.1 SCalSpecFluxConv

Meta Data:

- It should have a metadata called "biasMode."
  The flux conversion factors will depend on whether the data are taken in the nominal or bright-source mode.
- It should have a time stamp for its critical dependence on the NonLinCorr calibration product.
  Since the nonlinearity module has the freedom of adopting an arbitrary scaling factor in defining the parameters K_1 and K_2, any change to the NonLinCorr product would have an impact on the flux conversion calibration product. We should implement something like what is being implemented in the temperature drift module (which also critically depends on the NonLinearity calibration product).

Table Data

- Need units for each column.
- Need errors for each column as well.

3.4.2 ScalSpecOptCross

Meta Data:

- Is the metadata "BiasMode" needed? Probably not?

Table Data

- Since this is a matrix of channel vs. channel, we may need an error TableDataset per SSW/SLW?

3.4.3 ScalSpecBeamProf

Not in existence yet.

3.5 SxRs Raised

SxR number	Title
SPIRE SCR-1373	SciVal-FTS3: Give editions of calibration products meaningful names
SPIRE SCR-1374	SciVal-FTS3: SCalSpecInterRef to allow for four different resolutions
SPIRE SPR-1375	SciVal-FTS3: SCalSpecInterRef is difficult to read for humans
SPIRE SPR-1376	SciVal-FTS3: ChannelCalibTable should not cover N channels
SPIRE SCR-1388	SciVal-FTS3: Change errors in SCalSpecSmecZpd from 0 to 15nm
HCSS SPR-6676	MapContext presents products in non-alphabetical order

4 User Guide and Usability

4.1 User Guide: Successful Commands

An investigation into whether the commands as written in the User Manual executed successfully. This section lists the calls that executed successfully, and the following one lists the commands which were not successful and which should be updated in the User Manual.

4.1.1 engConversion

```
level0_5 = engConversion(obs.level[“level0”],cal=obs.calibration,
useSink=useSink)

block=obs.level[“level0”].get(obsid,bbid)
level0_5 = engConversion( level0block=block, cal=obs.calibration,
useSink=useSink)
```

4.1.2 First Level Deglitching

```
sdt = deglitch(sdt)

sdt=deglitch(sdt, scsaleMin=2,scaleMax=8,voices=5, thresholdHolder=-
0.6,thresholdCorr=0.985)
```

4.1.3 Electrical Cross Talk

```
sdt = elecCross(sdt, table=obs.calibration.spec.elecCross)
```

4.1.4 Nonlinearity Correction

```
sdt = specNonLin(sdt, table=obs.calibration.spec.nonLinCorr)
```

4.1.5 ClipCorrection

```
sdt = clipCorrection(input=sdt)

from herschel.spire.ia.pipeline.spec.clip import ClippingTask
clipping = ClippingTask()
clip = clipping(input=sdt, number=5)
```

4.1.6 Time Domain Phase Correction

```
sdt = timeDomainPhaseCorrection(sdt=sdt,
lpfPar=obs.calibration.spec.lpfPar,chanTimeConst=obs.calibration.spec.chanT
imeConst)
```

4.1.7 Temperature Drift

```
sdt = tempDrift(data=sdt,table=obs.calibration.spec.tempDriftCorr)
```

4.1.8 Compute BSM Angles

```
bat = calcBsmAngles(nhkt, bsmPos=obs.calibration.spec.bsmPos)
```

4.1.9 Create Interferograms

```
sdi = createIfgm(sdt=sdt, smect=smect, hkt=nhkt, spp=spp,
calSmecZpd=obs.calibration.spec.smecZpd,
calSpecChanTimeOff=obs.calibration.spec.chanTimeOff,
calSpecSmecStepFactor=obs.calibration.spec.smecStepFactor,
interpolType=”spline”)
```

4.1.10 TelScalCorrection

```
sdi=telScalCorrection(sdi=sdi,
sdical=obs.calibration.spec.interRef,nhkt=nhkt)
```

4.1.11 Baseline Correction

```
sdi = baselineCorrection(sdi=sdi, type='polynomial', degree=4)

sdi = baselineCorrection(sdi=sdi, type='fourier', threshold=4)
```

4.1.12 Second Level Deglitching

```
sdi = deglitchIfgm(sdi=sdi, deglitchType = "MAD_WINDOW", windowSize=33)
```

4.1.13 Double/Single Sided Fourier Transform

```
dsds = fourierTransform(sdi=presdi, ftType="ds", IA=True)

dsds = fourierTransform(sdi=presdi, ftType="ds", IA=False)

dsds = fourierTransform(sdi=presdi, ftType="ds", IA=True, ZP=True)

ssds = fourierTransform(sdi=sdi, ftType="ss", IA=False)
```

The parameter IA seems unnecessary here; according to the manual it only allows the user to start using the ZP parameter. It also isn't clear what type of zero-padding is done, if one has IA = True and then doesn't specify the ZP condition.

4.1.14 Phase Correction

```
sdi = phaseCorrection(sdi=sdi, sds=dsds, polyDegree=4, pcfSize=127,$
bandEdge=obs.calibration.spec.bandEdge)
```

All apodization functions for the parameter "convolApodName" as listed in the User Manual were acceptable functions.

4.1.15 SpecFluxConversion

```
ssds = specFluxConversion(sds=ssds, fluxConv=obs.calibration.spec.fluxConv)
```

4.1.16 SpecOptCrossCorrection

```
ssds = specOptCrossCorrection(sds=ssds,
optCross=obs.calibration.spec.optCross)
```

4.2 User Guide: Failures

In each case the code that was attempted to be executed is shown, followed by the error message that it produced.

4.2.1 engConversion

The Engineering Conversion Pipeline Step-by-Step

```
rpdt = obs.level["level0"].get(obsid,bbid).rpdt
pdt = formatConversion(rawData=rpdt)

……….….herschel.ia.task.SignatureException:rawData: Null is not allowed
```

4.2.2 First Level Deglitching

Example

```
from herschel.spire.ia.pipeline.common.deglitch import DeglitchingTask
deglitch = DeglitchingTask()
sdt = deglitch(sdt,scaleMin=2,scaleMax=8,voices=5,hmin=-
1.3,thresholdHolder=-0.6, $
 thresholdCorr=0.985)
```

```
…………..herschel.ia.task.SignatureException: The parameter with name hmin is
not present in the Signature
```

4.2.3 Temperature Drift

No manual notes found.

4.2.4 Create Interferograms

Example

```
from herschel.spire.ia.modules.ifgm import *
```

```
………….Import Error: No module named modules
```

4.2.5 TelScalCorrection

```
sdi = telScalCorrection(sdi=sdi, sdical =
obs.calibration.spec.interRefList.getProduct(
sdi.meta['biasMode'].value, sdi.startDate), nhkt=nhkt)
```

```
………….Attribute Error: 'None' object has no attribute 'getProduct'
```

Example

```
sdi = telScalCorrection(sdi=sdi, sdical=cal, nhkt=nhkt)
```

```
……….....NameError: 'cal'
```

4.2.6 Second Level Deglitching

Example

```
sdi = deglitchIfgm(sdi=sdi, deglitchType="STDDEV", windowSize=33)
```

```
………….deglitchType="STDDEV" should be replaced with deglitchType="STD";
other options include "MAD", "STD_WINDOW" and "MAD_WINDOW"
```

4.2.7 Interferogram Apodization

The parameter 'apodFunctionName' can be any of a number of possible functions. The following were found to be invalid/not accepted by HIPE:

- aNB_W_120 (Weak Norton Beer)
- aNB_M_140 (Medium Norton Beer)
- aP_141
- aNB_S_160 (Strong Norton Beer)
- aBH_3_184 (Blackman Harris, 3 terms)
- aE_195 (Filler E(α) function with α=0.22
- aBH_4_221 (Blackman Harris, 4 terms)

whereas the following were accepted by HIPE/gave the expected result:

- aNB_11 – aNB_20
- aHANN (Hanning)
- aHM_150 (Hamming)
- aGAUSS (Gaussian)

4.2.8 Average Spectra

```
asds = averageSpectra(sds=ssds, dir=”up(down)”)
```

```
………..herschel.ia.task.SignatureException: Error in parameter with name dir:
value up (down) with type org.python.core.PyString is incompatible with
type java.lang.Boolean
```

4.2.9 Additional Points

The ObsID 0x30011904 was independently run through the SOF2_pipeline.py script without event.

Depending on whether or not the final order of the modules has been decided upon, this ordering should be synced with that of the module descriptions in the User Manual.

4.3 Module & Keyword Naming

A review of the names used for modules and keywords was carried out. The following recommendations reflect improvements which will aid the user in understanding each module, and also recommendations based on consistency with the Software team Pipeline Policies.

Following the TWiki page, http://www.herschel.be/twiki/bin/view/Spire/SpirePipelinePolicies, revision 16 from June 4, 2009, the following General Task Policies relating to naming are defined:

From Appendix C of the Developer's Manual (HERSCHEL-HSC-DOC-0625, 14 October 2008):

Task (*UM chapter 8*)	A Task is a class which can be called as a function. Tasks do input and output parameter type checking and provide history to Products.	Names follow the same conventions as for classes. Task names should end with the word 'Task'. DisplayDataFrameTask, ResampleTask

The format for the names of task parameters shall depend on the parameter's type of task parameter. That is, one format shall be adopted for data products, one format for calibration products, one format for control parameters, and one format for keyword inputs.

TaskParameter Type	**Format**	**Example**
Data Product (e.g. PhotometerDetectorTimeline, SpectrometerDetectorSpectrum, SpectrometerMechanismTimeline, NominalHousekeepingTimeline)	The value in the product's type MetaData, all lowercase	SpectrometerDetectorSpectrum, use sds
Calibration Product (e.g. PhotPcal, SpecChanTimeConst)	A camelCase string that represents the product and its location in the calibration context	obs.calibration.spec.chanTimeConst, use chanTimeConst
Control Parameters (e.g. Interpolation type, Apodization function)	A camelCase string	e.g. interpolType, ftType, apodFunctionName
Keyword Parameters that are of type boolean (e.g. Usage of IA mode)	An uppercase string	e.g. IA in the FourierTransformTask

Order	TaskParameter Type	Note
1	Data Product	If the task can accommodate more than one data product, the first in the list shall be the Primary input product, followed by **Mandatory** products, then **Optional** products
2	Calibration Product	**Mandatory** calibration products followed by **Optional** calibration products
3	Control Parameters (e.g. Interpolation type, Apodization function)	Sort Alphabetically
4	Keyword Parameters that are of type boolean (e.g. Usage of IA mode)	Sort Alphabetically

We recommend the following changes to the General Task Policies:

- The official pipeline scripts should not use alternatives to the standard task names when instantiating them in the script. The instantiated name should be the task name without "Task" at the end - for example, for the temperatureDriftCorrectionTask, the instantiated name should be temperatureDriftCorrection.

- The pipelines should consistently specify the first argument of the task. Preferably without requiring specifically typing the keyword name - i.e. temperatureDriftCorrection(sdt) instead of temperatureDriftCorrection(sdt=sdt). Currently this is not consistent in the pipeline script (it may only require a change in the pipeline script to implement).

The policy about Boolean keywords is not universally followed at the moment (e.g. the "useSink" keyword of the engineering conversion). We think it would be preferable to adjust the policy to allow this rather than to update this particular keyword. For example changing the rule to:

Keyword parameters of type Boolean **can** use all UPPERCASE letters instead of camelCase

We recommend the following specific changes to task names, keyword names, and documentation (the current name of the task and the name currently used to instantiate it in the pipeline script are given in bold):

EngConversionTask() – *engConversion*
It would be more self-explanatory to rename the keyword "useSink" to be "tempStorage".

CalcBsmAnglesTask() – *calcBsmAngles*
We recommend that the calcBsmAngles task and the SpirePointing product should be moved later in the pipeline script to be as close to the CreateIfgm task as possible (this is where the pointing product is used).

SpirePointingProduct
The product should use keyword, "detAngOff", to accept the calibration product (as described in the Pipeline Policy).
Shouldn't there be a task called "SpirePointingTask" that returns an instance of the SpirePointingProduct, rather than using a product constructor directly? It would probably be less confusing to Astronomers if it was called in the same way as the other tasks (with a lower case letter first).

ElecCrossTask() – *elecCrossTask*
The task should use keyword, "elecCross" to accept the calibration product (as described in the Pipeline Policy).

DeglitchingTask() – *deglitch*
The current instantiated name does not match the task name at the moment. We recommend that the task name should change to the more self-explanatory "deglitchTimelineTask". This will distinguish it from the second level deglitching task which does not work on timelines (and is called "deglitchIfgmTask"). The instantiated name in the pipeline script should then be "deglitchTimeline".
We recommend updating the keywords to make them more understandable and self consistent:

- Change keyword "voices" to "scaleInterval"
- Change keyword "hMin" to "holderMin"
- Change keyword "thresholdHolder" to "holderMax"
- Change keyword "thresholdCorr" to "correlationThreshold"

SpecNonLinearityCorrectionTask() – *specNonLinearityCorrection*
The task should use keyword "nonLinCorr" to accept the calibration product (as described in the Pipeline Policy).

TemperatureDriftCorrectionTask() – *temperatureDriftCorrection*
The task should use "tempDriftCorr" to accept the calibration product (as described in the Pipeline Policy).

ClippingTask – *clippingCorrection*
We recommend that the task name should be updated from "ClippingTask" to be "ClipplingCorrectionTask".
The keyword, "number", should be updated to make it more understandable. We recommend it is changed to "fitPoints"

CreateIfgmTask() – *createIfgm*
Remove the explanation about over-sampling from the User Guide documentation if that option is not available.

ScalTask() – *telScalCorrection*
We recommend to change the task name from "ScalTask" to "TelescopeScalSubtractionTask" and to update the instantiated name in the pipeline script to match.
The task should use keyword "referenceIfgm" to accept the calibration product (as described in the Pipeline Policy).

ApodizeIfgmTask() – *apodizeIfgms*
The instantiated name in the pipeline script should be changed from "apodizeIfgms" to "apodizeIfgm".
We recommend to change the keyword name "apodType" to "ifgmType" (to be consistent with the Fourier transform task) and to change accepted values of the "ifgmType" keyword from "ds"/"ss" to "doubleSided"/"singleSided"
We recommend to change the keyword "apodFunctionName" to "apodName", and also to change the names of the accepted apodizing functions to the following pattern: a unique description of the function in camelCase; an underscore; two numbers signifying the factor by which the instrumental line shape is broadened (16 if that factor is 1.6). An example is "adjustedNortonBeer_16".

FourierTransformTask() – *fourierTransform*
We recommend to change the keyword name "ftType" to "ifgmType" (to be consistent with the Apodization task) and to change possible values of the "ifgmType" keyword from "ds"/"ss" to "doubleSided"/"singleSided".
We recommend to replace the two logical keywords "IA" and "ZP" with one keyword called "zeroPad", with three possible values: "none", "standard" and "2**n".
The difference between "standard" and "2**n" zero-padding should also be explained more clearly in the User Guide.

PhaseCorrectionTask() – *phaseCorrection*
We recommend to change the keyword "sds" to "doubleSidedSpectrum" for the secondary input. This will make it much clearer what this input is supposed to be.

AverageSpectraTask() – *averageSpectra*
We recommend to change the keyword "dir" to "direction" and allow only the values "forward" or "reverse".

4.4 SxRs Raised

SxR number	Title
SPIRE SCR-1645	SciVal-FTS3: Recommendations to improve consistency of pipeline task and keyword naming (for FTS pipeline scripts)
SPIRE SCR-1646	SciVal-FTS3: Recommendation to improve naming in EngConversionTask()
SPIRE SCR-1647	SciVal-FTS3: Recommendation to improve naming in CalcBsmAnglesTask()
SPIRE SCR-1648	SciVal-FTS3: Recommendation to improve naming in SpirePointingProduct
SPIRE SCR-1649	SciVal-FTS3: Recommendation to improve naming in ElecCrossTask()
SPIRE SCR-1650	SciVal-FTS3: Recommendation to improve naming in DeglitchingTask()
SPIRE SCR-1651	SciVal-FTS3: Recommendation to improve naming in SpecNonLinearityCorrectionTask()
SPIRE SCR-1652	SciVal-FTS3: Recommendation to improve naming in TemperatureDriftCorrectionTask()
SPIRE SCR-1653	SciVal-FTS3: Recommendation to improve naming in ClippingTask()
SPIRE SCR-1654	SciVal-FTS3: Recommendation to improve naming in TimeDomainPhaseCorrectionTask()
SPIRE SCR-1655	SciVal-FTS3: Recommendation to improve naming in ScalTask()
SPIRE SCR-1656	SciVal-FTS3: Recommendation to improve naming in ApodizeIfgmTask()
SPIRE SCR-1657	SciVal-FTS3: Recommendation to improve naming in FourierTransformTask()
SPIRE SCR-1658	SciVal-FTS3: Recommendation to improve naming in PhaseCorrectionTask()
SPIRE SCR-1659	SciVal-FTS3: Recommendation to improve naming in SpecFluxConversionTask()
SPIRE SCR-1660	SciVal-FTS3: Recommendation to improve naming in SpecOptCrossCorrectionTask()
SPIRE SCR-1661	SciVal-FTS3: Recommendation to improve naming in AverageSpectraTask()

5 ERROR PROPAGATION

5.1 Summary of Activities

The science validation group prepared a document (Error Propagation in the SPIRE Spectrometer Data Processing Pipeline) to identify the uncertainties at each stage of the pipeline. The Fourier Transform task was identified as the critical step where error propagation presents an unsolved problem. Currently, error estimates enter the data processing pipeline at the AverageSpectraTask where the standard deviation as a function of frequency is entered into the error columns of the resulting spectra. The existing calibration and data products were reviewed in terms of their ability to propagate errors and respective software changes requests were issued.

5.2 SxRs Raised

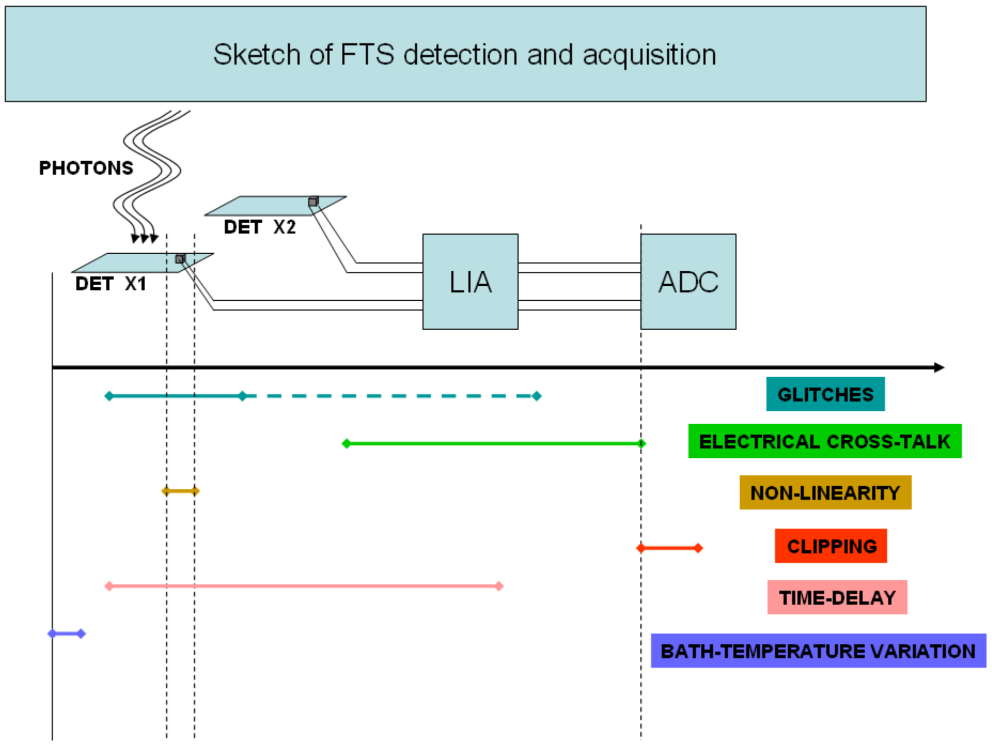
SxR number	Title
SPIRE SCR-1735	SciVal-FTS3: Provide placeholders to propagate error information
SPIRE SCR-1736	SciVal-FTS3: error columns for raw data timelines
SPIRE SCR-1737	SciVal-FTS3: error columns for SpecChanGain
SPIRE SCR-1738	SciVal-FTS3: error columns for the Spectrometer Detector Timeline
SPIRE SCR-1739	SciVal-FTS3: error columns for the BSM Timeline Product
SPIRE SCR-1740	SciVal-FTS3: error columns for SMEC Timeline Product
SPIRE SCR-1741	SciVal-FTS3: error columns for SpecElecCross
SPIRE SCR-1742	SciVal-FTS3: provide covariance matrix to the non-linearity correction task
SPIRE SCR-1743	SciVal-FTS3: error columns for SpecFluxConv
SPIRE SCR-1744	SciVal-FTS3: error columns for SpecOptCross
SPIRE SCR-1745	SciVal-FTS3: error columns for SpecBeamProf

6 APPENDIX A: SEQUENCE OF TIMELINE MODULES

This short document lists the relevance and consequences of having a specific sequence in the pipeline modules. First a rather simplistic physical consideration on the temporal sequence with which events occur in the instrument and subsequently a list of specific (albeit sometimes very particular) cases where problems will arise in reconstruction.

6.1 Consecutio Temporum

The idea is to point out the physical causality of individual effects, and propose a new sequence for these modules for further discussion and testing in phase 3 of the validation process.
If we represented the events that take place from the moment that the photons reach the detector up to the data collection of a detector timeline we see the following:



The current version of the pipeline and the way it removes/corrects for the different effects is represented by reading the sequence of modules on the right from top to bottom.

Reading the left side of the diagram in the direction of the time arrow one could say the following.

a) The photons arrive on the detector (absorber) and "find it" at its temperature (initially dictated by bath fluctuations)

b1) At this point photons are absorbed with a given efficiency producing a variation in the temperature of the detector. This variation occurs with a time-delay function of the thermal coupling detector-absorber.

b2) We need also to mention that from this moment onwards, cosmic rays can hit the absorber or the detector producing an additional (spike-in-nature) heating which will still be affected by the combined thermal time constants of absorber+thermistor.
Glitches can also impact wires or parts of the readout electronics producing again signal-spikes that this time are not affected by thermal time delays but from any time-delay associated with the down-stream electronics.

c) As the temperature variation reaches the thermistor it is transduced to the resistance where any non-linear-behaviour occurs.

d) At this point the signal current is carried by the detector wiring to the cold JFET stage. During this stage, wiring issues/architecture of different detectors can induce cross-talk (of electrical nature)
e) Finally the signal reaches the ADC where it gets converted to numbers in a specific range and where the this range is not suitable for the voltage level, clipping occurs.

Ideal reconstruction from the digitised signal should reverse the above list in a manner as close as possible to a time-inverted sequence in order not to mix causality of events and avoid creation of non-existent features.

However, the physical processes, due to their nature, are not completely sequential and can sometimes be considered independent.
The following sections describe a few (particular but not impossible) cases which could occur or which could present challenges in their correction.

Bearing in mind that:

- These might have already been considered and discarded for ease of pipeline programming
- The effects and cases that we consider here might be very small or negligible but should be quantified in order to justify the different module sequence

6.2 Glitches → Electrical cross-talk

Scenario: Glitches affecting the electrical crosstalk correction.

Currently deglitching is the first module applied and so glitches are removed from the timeline prior to the electrical cross-talk correction. However, if a glitch is sufficiently strong to be propagated by electrical crosstalk to another detector (after all this is how the correlation matrix is currently calculated), its contribution to that detector would not be removed. In fact, if the correlation matrix is symmetric, the correction would propagate the glitch back to the original detector from which had already been removed.

Consequence: A residual population of crosstalk-induced glitches below the threshold detection contaminates some of the channels.

6.3 Glitches → Clipping

Scenario: Clipped glitches affecting the Clipping correction.

A clipped glitch will show as a single clipped sample (maybe more?). The effect of the deglitching and clipping modules on each other depends on what they each do with samples flagged as "truncated" or "glitched". Does the clip flag prevent the de-glitcher from acting? Does the glitched flag prevent the clipping correction from trying to carry out its 8th order fit.

Current questions are:

1. Clipped glitch is flagged and removal is attempted. Is the wavelet reconstruction algorithm mislead by the truncated value?
2. Clipped glitch activates the clipping correction module on
   a. An unremoved feature. This could lead to possible artefacts in the interferogram?
   b. Removed feature (timeline with noise). What is the 8th order polynomial going to create? The deglitching task would have to modify the truncation flag if it removed a clipped glitch.

6.4 Electrical cross-talk → Clipping

Scenario: Electrical crosstalk propagating clipped samples.

If there is clipped data on a correlated detector channel, this feature would be transferred to other detectors via the crosstalk matrix if clipping were not previously corrected.

Consequence: Creation of artificial features on the timeline.

6.5 Non-linearity & Time-domain phase correction

The two effects are time-coincident and also correlated. In the Spire Photometer model, the time constant of the detectors and their response (linear or non-linear) is calculated at every step making the two corrections somewhat inseparable. There is no reason for considering one module prior to the other, but it is potentially dangerous to have other modules between the two.

6.6 Other “unsolvable” questions

PH in CLIPPED REGION:
If a particle hit occurs while in the ZPD region on a detector where the baseline of the interferogram is close to the ceiling of the ADC range and ZPD is clipped, the glitch would be completely hidden out of the dynamic range. However, electrical cross-talk may still propagate the glitch to other detectors, even though the glitch itself and its effects are hidden in the clipped region of the original detector timeline. This would not be corrected by the electrical crosstalk module as the Clipping correction cannot reconstruct the glitch. Hence the correlated timelines have to live with the additional noise spike.

NON-Linear in CLIPPED REGION (if CLIPPING is first module):
If non-linear behaviour occurs in a clipped region, after the Clipping correction restores the interferogram, this does not include the Non-linearity behaviour as if the IG hadn’t been clipped. The NL algorithm would then attempt to correct for the non-linearity based on detector parameters and over compensate.

6.7 Conclusion

The physical arguments presented above indicate that the order of the timeline modules should be:

1. Clipping
2. Electrical Crosstalk
3. Deglitching

4a. Non-linearity
4b. Time Domain Phase Correction
6. Bath temperature fluctuations

7 APPENDIX B: SOME CONSIDERATION OF THE MODULES SEQUENCE BY THE LAM TEAM (JPB, CHS, DBG)

with modifications (JPB, May 4th)

7.1 Temporal domain

7.1.1 1st Level Deglitching

A priori should be placed anywhere in the temporal domain.

7.1.2 Remove electrical Crosstalk

For this module we need to have as input a timeline as closed as possible to the original analogical signal.
So the module should have as input an SDT « not corrected for glitches ». However a « clipped glitch » remains as an intrinsic problem since the actual power in this peculiar glitch is unknown (we only have a lower limit for it!) and therefore the crosstalk effect will be underestimated.
In case of a « clipped » signal near ZPD, in order to get the true signal level, we should need to have the clipped channel restored (as best as possible) as it should have been before clipping by the ADC device. This would require that the Clipping Correction module should be inserted before.

7.1.3 Non-Linearity Correction

This module is expected to restore the signal timeline as it should have been if recorded by a perfect (linear) detector.
The module is expected to provide the actual input power so, for all purposes requiring the actual power, it should be done first.

7.1.4 Clipping Correction

The « clipped » flag is already provided in the common Photometer/Spectrometer processing modules.
The module only corrects « clipped » samples within very peculiar conditions, i.e. when the number of consecutive « clipped » samples are less than 8 : this provides some degree of quality for the « reconstructed » samples.
These conditions exclude the cases of clipping at high-OPD and the cases of « clipped and glitched » samples as well since the reconstruction procedure is not appropriate for these two situations.
This is still TBC, but we think that the module reconstruction quality is increased with signals which have been corrected for Non-Linearity. If this statement is confirmed then the Non-Linearity Correction should be done prior to Clipping Correction.

7.1.5 Correct Time Domain Phase

As the output 'analogical' signal from any detector is filtered by the low-pass 'detector' and 'electrical' filters it seems reasonable to consider that this module should be done before any correction about non-linearity or detector response (bath temperature).
However, since the module is modifying any individual sample through a convolution process then it seems to make more sense to have this module at the end of the time domain sequence.
This is TBC if the module requires, for a better quality of the module output, that any « clipped » sample has to be 'reconstructed' or not before this step. As the clipped areas are close to ZPD where the signal has the highest contrast (and SNR) this should be of peculiar concern. So some tests should be done in order to investigate this issue. [Is such tests already done ?]
Also TBC the effect of glitches on the quality of the time domain correction. It seems reasonable to consider that the effect should not be so important except, maybe, the cases of 'clipped' glitches. [Has someone done any investigation on this issue too ?]

7.1.6 Bath Temperature Correction

The module is able to correct detector response changes smoothed at a frequency of a fraction of Hz. Again this module restores the actual power in the signal timeline (as does the Non-Linearity Correction), so it should be performed prior to any module requiring the actual power timeline..

7.1.7 Conclusions

From the considerations given above we suggest the module sequence to be as follows:

1. 1st Level Deglitching	only detection to get 'glitch' flags
2. Clipping Correction	restore the SDT as close as possible to the analogical signal
3. Remove electrical Crosstalk	using best information on analogical signal
4. 1st Level Deglitching	detection and reconstruction of 'glitched' areas
5. Bath Temperature Correction*	
6. Non-Linearity Correction*	
7. Clipping Correction	redone on corrected timeline for a better quality reconstruction
8. Correct Time Domain Phase	

Notes:
1. * steps 5 & 6 can be done in reverse order (I have no strong arguments to perform one module before the other one)
2. I could agree that steps 5, 6 and 8, as the associated effects are time-coincident and correlated, should not be separated. However the clipping correction which 'restores' only the 'clipped' samples should have no unwanted effects on the last module
3. Steps 1 & 2 can be dropped if the tests (see points 1.2 & 1.5 above) show that step 3 does not require them to be done first. Again step 7 can be dropped if the tests show no significant improvement in the clipping reconstruction after non-linearity correction.

As concerns the 'spatial' and 'spectral' domains there is no obvious rationale to propose a different scheme than in the present pipeline structure.

7.2 Spatial domain

1. SCAL & Telescope Removal
2. Baseline Removal
3. 2nd Level Deglitching
4. Phase Correction
5. Apodization

7.3 Spectral domain

1. Spectral Response & Flux Conversion
2. Remove Optical Crosstalk
3. Average Spectra