1 Introduction

This note intends to give a description of the calibration files needed to assign a coordinate on the sky to any PACS ‘elementary measurement’ in the sense of a signal measured at a given time, for a given photometer or spectrometer detector, at a given chopper position, and for the satellite pointing parameters applicable at that time. The intention is to give clear verbal descriptions of the content, but not IA implementation details.

This philosophy of assigning coordinates to all elementary measurements is to be contrasted with one where a coordinate is assigned to a reference pixel in an ‘image’ already at an early (frames) stage of PACS data processing, together with a world coordinate system (WCS) projection prescription how to convert this image to world coordinates. Such an approach appears to be less suited for PACS at the frames level for two reasons: (i) The individual bolometer ‘images’ are non-contiguous and, due to the specifics of matrix alignment and the width of the inter-matrix gaps, are not even subsets of a single regular pixel grid. (ii) Maps may often be reconstructed from a ‘naive mapping’ weighted averaging of all elementary measurements in a certain sky bin, with elementary measurements taken at different times and by very different detectors of the array. Such an approach implies frequent calculations of RA, DEC (for each measurement), and knowledge ‘which measurements are inside a certain region on the sky’, the region being defined by a pixel in a map of a certain projection, or a radius around a position. Similar requirements arise from map making algorithms like madmap that are being implemented. Proper WCS information is of course essential at the stage of the final maps or images. It should be noted, however, that an image/WCS approach remains possible for the bolometer section if each matrix separately is considered as an ‘image’. The WCS parameters could then be derived in a second step from the calibration data outlined below and allow interfacing with popular outside reduction packages. The image/WCS approach is not suited for the 5×5 spectrometer IFU ‘images’ because the alignment imperfections of the five slices cannot be captured in simple distortion parameters.

The note does not address spatial quantities like the PSF that need to be quantified and may be captured in ‘calibration files’, but are not directly part of the data processing from detector readouts to a map with proper astrometry. See PICC-ME-TN-033 for a description of the photometer PSF.

At certain points, it is based on preliminary assumptions e.g. on AOT design that have to be revisited once they are documented. Where possible, the conventions used in IA should not diverge too much from those used by the simulator (see ‘The PACS simulator’, PACS-SIM-Saclay-02).
2 Assumptions

2.1 Chopper calibration

It is assumed here that calibration files are available that allow the conversion of chopper readback to a physical chopper angle with respect to the instrument optical axis. For example, these could be two Calibration files:

1. A conversion function from chopper readback to chopper angle $\alpha'(CPR)$ [degrees]. ChopperAngle is the implementation of this calfile, and chopPos2FpuAnglePol and fpuAngle2ChopPosPol the associated IA conversion routines.

2. The Chopper readback value CPR of the instrument optical axis. In the spirit of PCD 2.3.1 the optical axis would be defined as the center between the chopper angles where a (fictitious for the bolometer) central detector runs into the two calibration sources, or alternatively the center between the chopper angles where two detectors that are symmetric to the array centers hit the calibrators. For FM-ILT3 we adopted CPR = 664 which is the position of the chopper ‘optical zero’ and, according to quick tests, in rough agreement with the ‘middle between the two calibration sources’ as defined above (DID SOME REPORT VERIFY THIS MORE ACCURATELY). This is currently (post SCR 1153) captured as ‘zeroOffset’ in the chopperSkyAngle calfile that is in the common branch.

The physical chopper angle $\alpha$ with respect to the optical axis at a given chopper position is then

$$\alpha = \alpha'(CPR) - \alpha'(CPR_0) \quad [\text{degrees}]$$

Post SPR xxxx, the angle filled into the Frames status column CHOPFPUANGLE is this angle $\alpha$ relative to optical zero.

2.2 Pointing

At the PACS AOT workshop in February 2005, solutions were discussed to implement the many different ‘aim points’ that emerge from the need of correctly pointing observations with different chop throws, observations with unchopped line scans etc. We preferred a solution with very few fixed ‘virtual apertures’ (one for photometer, one for spectrometer) the complexity being delegated to ‘offsets’ (in spacecraft Y,Z, and assumed to be in arcsec) that can be flexibly applied inside the AOT logic. In that system, the following quantities have to be known for IA at any time:

- The virtual aperture used. This is constant over a pointing request and should be available in the header.
- The RA, DEC, position angle (2000) at which the ‘virtual aperture’ is pointing instantaneously. The downlink pointing product has RA, DEC and Roll angle for the telescope ACF/ACA which then has to be converted to the virtual aperture using the SIAM.

N.B: The Y,Z offsets then have been applied by the AOT logic already in uplink. They are constant over an AOR by definition. As a consequence, the instrument boresight pointings in the POS file
'INFO RREQ' entry will already differ from the position typed into HSPOT for AOR modes that apply an offset. The RA, DEC at various stages in downlink always refer to ACF or virtual aperture coordinates that include the offset applied.

Note the ongoing discussion (HCSS SCR 5539) on whether chopper misalignment should be dealt with by an explicit skew parameter in the AOT logic or by rotating the SIAM and hence the direction in which offsets are applied.

2.3 Detector sorting

This bookkeeping issue is assumed to be solved here, i.e. the matrices/modules are seen in the right general sequence and orientation in IA, without need for additional mirroring or major ‘90/180/270 degree’ rotations.

2.4 Overall orientation detector relative to sky

This is affected by the PACS and Herschel optical scheme and the detector sorting in the PACS electronics and in IA. Rather than trying to trace all the related steps, the approach taken here is to rely on ILT measurements using the xy stage, combined with the relative orientation of various coordinate systems as based on optical design. From PACS-ME-TN-073 Issue 1, PACS-ME-DS-003 and additional priv. comm. (N. Geis and E. Wiezorrek) movements of a point source on the xy stage translate in the following way:

\[ +x_{\text{Stage}} \rightarrow -X_{\text{SAT}} \rightarrow +Z_{\text{TEL}} \rightarrow -Z_{\text{Sky}} \]
\[ +y_{\text{Stage}} \rightarrow -Y_{\text{SAT}} \rightarrow +Y_{\text{TEL}} \rightarrow -Y_{\text{Sky}} \]

Where the coordinate systems are:
- Stage: Commanded movement of the point source on the xy stage.
- SAT: Movement of the stage in the Herschel/PACS xyz coordinate system, expanded to include the full testoptics and xy stage setup. This is peculiar to the folding of the optical train in the ILT/OGSE setup and has no significance for the in-orbit situation. Don’t confuse with a different use of index ‘SAT’ in IID and Herschel Observer’s manual.
- TEL: Movement of the source in the Herschel telescope focal plane, in the Herschel xyz ‘SCA’ coordinate system.
- Sky: Movement of the source on the sky relative to the projected yz directions of the Herschel xyz coordinate system.

See also the overview sketch in the PICC-ME-TN-027 ‘cheat-sheet’.

**WARNING:** Versions of this document up to 0.6 had a different sign of the conversion Stage to Telescope in the Y direction. This was based on PACS-ME-DS-003 and is superseded by PACS-ME-TN-073!
Figure 1: Subarray coordinate system $p,q$. These (integer) coordinates for the centers of pixels are the ones used in the bolometer frames and in their IA display. The orientation of this figure is as in the IA Display.

3 Bolometer: Definition of coordinate systems

Following PCD section 3.1, we define the following coordinate systems:

3.1 Subarray coordinates $p,q$

A coordinate system for each subarray, trivially assigning the center of each pixel with an integer coordinate e.g. $p,q=(4,5)$. For simplicity, we do not define separate subarray coordinate systems for each matrix but adopt the single one that is implicit in the indices of the bolometer frames class in IA and that is the one seen in an IA ‘Display’.

The optical inversion between red and blue section of the bolometer is now (FM-ILT) already considered in the hardware/detector sorting, i.e. it does not have to be addressed explicitly by IA software.

3.2 Array coordinates $u,v$

A cartesian coordinate system in the PACS focal plane, with the long ‘v’ axis aligned with the chop direction. The signs are chosen such that orientation (aside of the few degree misalignment) is the same for $p$ and $u$, and for $q$ and $v$. This implies that a fixed source will apparently move in positive $v$ direction over the array when the chopper is commanded to more positive CPR values. The zero point of this coordinate system is placed at the center of each array (blue/red), defined by the center of gravity of all pixels of this array (including bad/dead ones). This implies a slight offset of this zero point between red and blue, when projected onto the sky (This convention was adopted starting from FM_2.0 calfiles)
Figure 2: Array coordinate system u,v. Positions of pixels centers for the various blue matrices are schematically indicated, exaggerating true misalignments. The v direction is aligned with the chop direction.

3.3 Instrument coordinates y,z

An orthogonal local coordinate system on the tangential plane of the sky. The two axes are aligned with the spacecraft Y and Z axes as projected onto the sky, and the zero point is the ‘optical axis’ of the bolometer in the sense that it corresponds to the zero point of the array coordinate system for the blue array and chopper angle \( \alpha = 0 \). This zero point is defined as the PACS bolometer virtual aperture which is directly used for line scanning. The other modes are dealt with in the AOT logic by using the same virtual aperture and Y,Z offsets.

The coordinate axes of this system are parallel to those of the familiar ‘PACS FOV Footprint’ Diagram (IDD Part I Issue 3.1 Figure 4.2-1, note this is the view from above the focal plane). The zero point (bolometer virtual aperture) is shifted with respect to the Herschel optical axis by roughly 80mm in positive z direction in the focal plane, corresponding to roughly 10 arcmin in negative z projected onto the sky, and perhaps a small amount (depending on alignment) in y. Note the inversions between u and z and between v and y (Figs. 2 and 3).

There is a peculiarity in defining the zero point of both instrument and array coordinates on a point that is not filled on the array. Given how we are setting up the photometry AOT with nod orthogonal to chop (SAp-PACS-MS-0186-03 v2.0), this may not be a problem, and consistent with the ‘central pointing position’ approach of PCD 3.1.1. If \( Y_{\text{chop}} \) is the chop throw in arcsec and \( Z_{\text{nod}} \) is the nod throw in arcsecond (assuming chopper and y were aligned, the measured chopper-y misalignment can be trivially considered), one can define the offset for such an observation as \( 0.5 \times Z_{\text{nod}} \) (assuming the chop is symmetric to the optical zero and nodding for this AOT is done by beamswitch, which will need such an offset towards the first nod position). PCD requirement 3.1.1 would then take the form of changing the XY positions of the virtual aperture until the 4-beam pattern is symmetric to both major and minor array axis of PACS.

The PCD till draft 7 used the term ‘telescope coordinates’ for a similar concept. ‘Instrument’ co-
Figure 3: Instrument coordinate system y,z as seen on the sky. For satellite position angle 0, z will point north and y east. The schematic indications for chop direction and for the bolometer FOV at different chopper elongations indicate possible misalignments and distortions. Distortion will depend on chopper position. Here, ‘chop direction’ indicates the direction the projected FOV will move on the sky when setting the chopper to more positive CPR values.

ordinates (From PCD draft 8, as also used by the simulator) may be a more proper term avoiding confusion between spacecraft XYZ coordinates and something that is specific to the PACS bolometer. To be precise, one should distinguish ‘Bolometer instrument coordinates’ and ‘Spectrometer instrument coordinates’ where necessary.

3.4 Sky coordinates

The J2000 ICRS right ascension and declination of an elementary measurement.

4 Bolometer: Transformations and associated calibration files

4.1 From subarray to array coordinates

We assume here that the subarrays are perfect quadratic grids, and that the conversion from subarray to array coordinates is solely driven by the scale (mm/pix) of these grids, their relative offsets, and their rotation.

Clarity might then suggest to define the corresponding calfile as scale, offsets, and rotation for each of the matrices. It is conceptually simpler and avoids repetitive calculation, however, to just delegate that to the calfile derivation and documentation and simply include in the calfile one step further down the u,v coordinates for the center of each pixel.

During the pipeline design it became obvious that coordinates of the pixel corners would be needed frequently enough to warrant inclusion in the calfile rather than repetitive calculation. This is the case since format version 2.0, starting with calfile version 2.1. The corners here are the corners of the $640 \times 640 \mu m^2$ active region inside the $750 \times 750 \mu m^2$ pixel.
The calfile design is hence:

**Calfile:**

\[ u, v \text{ values for the center of each pixel } p, q. \text{ These are simple two-dimensional arrays containing the } u \text{ and } v \text{ values as a function of the subarray coordinates } p, q. \text{ Two arrays for the blue bolometer section and two arrays for the red bolometer section.} \]

As of format version 2.0 (calfile version FMv4, there are, in addition to these four arrays, \(4 \times 4\) more arrays giving the same information for the four corners of each pixel (upper/lower, left/right).

Such a calfile definition would also be robust if the individual matrices were less ideal, or if crosstalk effects shift effective centroids and are not dealt with in a different way. Note that, due to the different image scale of red and blue camera, same \(\Delta u, v\) does not mean same distance on the sky for the two cameras.

There are two possible approaches for the derivation of this calfile: Component-level metrology of the relative positions of the matrices (PCD 1.1.15, available for FM blue, K. Okumura priv. comm.), and analysis of PCD 3.1.2 and 3.1.3. The latter two are mixed with optical distortions, with some ambiguities in separating ‘jumps’ between matrices (corresponding to the matrix misalignments addressed here) from larger scale smooth distortions (which are mostly optical). As described in the FM ILT analysis report (sect. 3.1), this separation was successful in practice.

This calfile is called PCalPhotometer\_SubArray\_Array\_version.fits (old system PacsCal\_Subarray\_Array\_version.fits) with the following versions:

<table>
<thead>
<tr>
<th>Version</th>
<th>Old</th>
<th>Date</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>QMv1</td>
<td>CQMv1</td>
<td>2005-06-22</td>
<td>Test version with nominal design values</td>
</tr>
<tr>
<td>FMv1</td>
<td>FMv1</td>
<td>2007-03-05</td>
<td>Preliminary FM-ILT version based on data from Dec 22, 2006</td>
</tr>
<tr>
<td>FMv2</td>
<td>FMv2</td>
<td>2007-10-11</td>
<td>Full FM-ILT3 version based on data from June 18/19, 2007</td>
</tr>
<tr>
<td>FMv3</td>
<td>FMv2</td>
<td>2007-10-18</td>
<td>Erroneous: Format version 2.0, but contents still the one of v2</td>
</tr>
<tr>
<td>FMv4</td>
<td>FMv2</td>
<td>2008-05-08</td>
<td>Format 2.0 with corner coordinates, otherwise identical to v2</td>
</tr>
<tr>
<td>FMv5</td>
<td>FMv2</td>
<td>2009-03-13</td>
<td>Added activeFraction meta (SPR1315), otherwise unchanged</td>
</tr>
<tr>
<td>FMv6</td>
<td>FMv2</td>
<td>2013-06-02</td>
<td>Full in-orbit rederviation (see PICC-ME-TN-044)</td>
</tr>
</tbody>
</table>

Here and in similar tables further down we list both the calfile version as used in the ‘new’ calibration framework starting mid 2008, and for completeness the equivalent ‘old’ versions that may be referred to in older documents or scripts.

The FMv1 version is obsolete and superseded by FMv2 and FMv6.

### 4.2 From array coordinates to instrument coordinates

This is clearly the most complex step of the PACS spatial calibration. Zemax optical modelling indicates optical distortions at a level that is relevant for data analysis, and that depends on chopper position.

We make the basic assumption that the transformation from array to instrument coordinates can be approximated by two polynomials in the three-dimensional space of \(u, v, \text{ and chopper angle } \alpha\).

\[
y = \sum_{i=0}^{N} \sum_{j=0}^{M} \sum_{k=0}^{O} a_{ijk} u^i v^j \alpha^k
\]
\[ z = \sum_{i=0}^{N} \sum_{j=0}^{M} \sum_{k=0}^{O} b_{ijk} u^i v^j \alpha^k \]

Where the units are:
- \( y, z \) [arcsec]
- \( u, v \) [mm]
- \( \alpha \) [degree]

The number of coefficients \( a \) and \( b \) needed, and in particular the number of significant ‘mixed’ terms, is not yet fixed. Preliminary modelling by Norbert Geis suggests that up to 3rd order in \( u \) and \( v \) may be needed, and that some mixed \( u, v \) terms are already needed from optics properties alone. In practice, the FM-ILT test report shows good accuracy already up to second order which was adopted for simplicity. It should be noted that because significant misalignments array/chopper\( /y \) are observed in the FM, mixed terms will occur by default. Any misalignment chopper/spacecraft \( y \) is implicit in the coefficients, there is no dedicated calfile with a misalignment angle. The current calfile design is robust to changes of the maximal polynomial order, if such are found necessary later.

**Calfile:**

*With separate entries for blue, green and red:*

For both \( y \) and \( z \):

- **Maximal orders \( N, M, O \) to be considered**
- \((N+1)^r(M+1)^s(O+1)\) polynomial coefficients

For ILT, we propose to define a calfile of the same format but with \( y, z \) interpreted in mm as \( y, x \) offsets of the \( xy \) stage for a centered pinhole. A zero order CQM ILT calfile implementing just nominal scale factors and offsets is the first exercise.

The derivation of this calfile is based on PCD 3.1.2 and 3.1.3

Because of different zero points, scale factors and also potentially different distortions, blue (then for both 70, 100\( \mu \)m) and red were separated in this calfile right from the beginning. After discovery of small residual offsets between blue/green/red in orbit, green was introduced separately from v6 on. This calfile is called PCalPhotometer_ArrayInstrument_version.fits (old system PacsCal_PhotArrayInstrument_version.fits) with the following versions:
**Figure 4:** When converting from instrument coordinates y,z to the actual sky RA DEC, the PA roll angle has to be considered (+20 deg in this example). The array footprint is schematically indicated for a nonzero chopper angle relative to optical zero, i.e. shifted relative to the virtual aperture. This sketch is not applicable near the celestial poles. The y,z offset relative to the position entered into hspot has already been applied in uplink.

### Table 1: Version History

<table>
<thead>
<tr>
<th>Version</th>
<th>Old</th>
<th>Date</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>QMv1</td>
<td>CQMv1</td>
<td>2005-06-22</td>
<td>Test version with nominal design values. y and z to be interpreted as xy stage y and x.</td>
</tr>
<tr>
<td>FMv1</td>
<td>FMv1</td>
<td>2007-03-05</td>
<td>Preliminary FM-ILT version based on data from Dec 22, 2006. y and z to be interpreted as xy stage y and x in mm. Chopper-related values nominal, not constrained by ILT.</td>
</tr>
<tr>
<td>FMv2</td>
<td>FMv2</td>
<td>2007-10-11</td>
<td>Full FM-ILT3 version based on data from June 18/19, 2007. y and z to be interpreted as xy stage y and x in mm.</td>
</tr>
<tr>
<td>FMv3</td>
<td>FMv2</td>
<td>2007-10-11</td>
<td>FM-ILT3-based ‘on-sky’ version. Distortion is the one for the xy stage case, but zeropoint and scale (in arcsec) adapted to the on-sky case. Has sign errors!</td>
</tr>
<tr>
<td>FMv4</td>
<td>FMv3</td>
<td>2008-12-08</td>
<td>FM-ILT3-based ‘on-sky’ version. Distortion has been transferred to sky using the model of PACS-ME-TN-073. Sign fix!</td>
</tr>
<tr>
<td>FMv5</td>
<td>FMv4</td>
<td>2009-10-12</td>
<td>First in-orbit version. Produced by rotating/shifting v4 for best match to OBSID 1342183563.</td>
</tr>
<tr>
<td>FMv6</td>
<td>FMv5</td>
<td>2010-06-14</td>
<td>Separated green from blue (PACS-2391, PACS-2617)</td>
</tr>
<tr>
<td>FMv7</td>
<td>FMv6</td>
<td>2013-06-02</td>
<td>Full in-orbit rederivation (see PICC-ME-TN-044)</td>
</tr>
</tbody>
</table>

FMv1 is obsolete. FMv2 and FMv4 are intended for parallel use when trying to interpret actual XY stage data (v2), or when needing an approximation of the real on-sky situation (v4). v3 is superseded by v4. Before the arrival of v5 an ad-hoc ‘sneak preview’ version was in use informally but never imported into the system. This is fully superseded by v5 and later versions.

### 4.3 From instrument coordinates to sky coordinates

These are not PACS-specific steps, and enumerated here only briefly.

1. Correct from projection on tangential plane to distance on sphere (can be omitted over our small field)
2. Compute RA, DEC for each measurement using the corrected y, z of the measurement, actual RA+DEC+position angle as available from the spacecraft data for the given time, and spherical trigonometrics (or using a simple ΔRA*cos(DEC) vs ΔDEC approximation if far from the celestial poles.

The relevant location of the related information in dp is (May 2010):

- The pointing product contains the quaternions representing the Herschel optical axis pointing (ACF=ACA=SCA). Note that at this point the downlink information is not yet corrected for aberration.
- The SIAM matrix containing the instrument/telescope alignment is defined as a calfile in dp-pacs. Contents of this file has to be generally consistent with the SIAM used in uplink, but holding it as a separate calfile rather than automatically using the uplink version allows to later apply minor tweaks/improvements at the processing stage.
- The pipeline step ‘AddInstantPointing’ reads the pointing product, applies the SIAM correction and the aberration correction and converts to RA, DEC, PA. The resulting information pertains to the virtual aperture and is stored in the ‘RaArray’, ‘DecArray’ and ‘PaArray’ columns of the frames status.
- The pipeline step ‘AssignRaDec’ uses the virtual aperture RA, DEC, PA, chopper angle and pixel coordinates to determine instantaneous RA and DEC for the center of each pixel, which are stored in the ‘Ra’ and ‘Dec’ cubes of the frames class.
- The pipeline step ‘PhotProject’ uses the same concepts to determine corner coordinates (not just centers) needed for its projection algorithm.

Calfile:

3 × 3 SIAM entries (direction cosines) for the two virtual apertures of the photometer (P01_0) and spectrometer (P02_0)

The content is a straight copy of an uplink SIAM, but cast into different form to (a) be consistent with the dp-pacs calfile system and its time-dependence mechanism and (b) allow to apply a different downlink siam in processing an observation than used in uplink of this specific observation.

This calfile is called PCalCommon-Siam_version .fits (old system PacsCal-Siam_version.fits) with the following versions:

<table>
<thead>
<tr>
<th>Version</th>
<th>Old</th>
<th>Date</th>
<th>ASCII SIAM</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>QM_v1</td>
<td>CQM_0</td>
<td>2008-02-05</td>
<td>0001_0006.SIAM</td>
<td>Test version, obsolete</td>
</tr>
<tr>
<td>FM_v1</td>
<td></td>
<td></td>
<td></td>
<td>Test version, obsolete</td>
</tr>
<tr>
<td>FM_v2</td>
<td></td>
<td>2009-06-01</td>
<td>0001_0009.SIAM</td>
<td>Pre-launch design, obsolete</td>
</tr>
<tr>
<td>FM_v3</td>
<td></td>
<td></td>
<td></td>
<td>Moved both apertures due to spire claims, obsolete</td>
</tr>
<tr>
<td>FM_v4</td>
<td></td>
<td>2009-06-29</td>
<td>0053_0002.SIAM</td>
<td>Moved both apertures post photometer sneak, obsolete</td>
</tr>
<tr>
<td>FM_v5</td>
<td></td>
<td>2009-07-23</td>
<td>0078_0001.SIAM</td>
<td>Moved only spectrometer. Valid pre-STR temperature change</td>
</tr>
<tr>
<td>FM_v6</td>
<td></td>
<td>2010-03-15</td>
<td>0320_0001.SIAM</td>
<td>Post-STR temperature change, Sign error, obsolete</td>
</tr>
<tr>
<td>FM_v7</td>
<td></td>
<td>2010-04-06</td>
<td>0320_0002.SIAM</td>
<td>Post-STR temperature change, obsolete</td>
</tr>
<tr>
<td>FM_v8</td>
<td></td>
<td>2010-04-12</td>
<td>0341_0001.SIAM</td>
<td>Post-STR temperature change, valid</td>
</tr>
</tbody>
</table>
“Post-STR temperature change” refers to the single OD 299 plus all ODs $\geq 320$. 
5 Spectrometer

A brainstorming session @MPE on 2007-10-12, based on the photometer experience and spectrometer ILT results, suggested a similar two-step implementation. The SubarrayArray step is here replaced by a conceptually similar ModuleArray step. The subarray coordinates p,q are replaced here by the module number m which is integer in the range 0 to 24.

The spectrometer has two possible additional dependencies of spatial calibration: From grating position and from spectral pixel within the module. The suggestion based on the FM-ILT results is to NOT consider these in the calfiles. There is very little dependence on grating position (i.e. grating alignment is good). Dependence from spectral pixel is measurable but small enough to ignore it for the sake of simplicity (see FM-ILT test report section 4.1).

5.1 From module to array coordinates

FM-ILT shows noticeable spatial misalignments between different ‘slices’ which suggests as first step a ‘ModuleArray’ calfile with positions for each module (=spatial pixel). The conversion from module to array coordinates captures the relative offsets of each module (=spatial pixel), with the chop direction as reference axis (v of the uv array coordinates) and the central module as zero point. The spatial scale is adopted as xxx mm per pixel (this scale effectively refers to the plane of the slicer). In analogy to the ‘SubarrayArray’ file, pixel corners should be included.

Calfile:

*u,v values for the center of each spectrometer module* . These are simple one-dimensional arrays containing the u and v values as a function of the module number m. Two arrays for the blue spectrometer section and two arrays for the red spectrometer section. 4×4 more arrays give the same information for the pixel corners.

The relative positions of the spectrometer modules involve a substantial part of the PACS optomechanical alignment. There is no simple component-level metrology to compare with, the calfile has to be based on analysis of ILT tests related to PCD 4.1.2 and 4.1.3.

This calfile is called PacsCal_ModuleArray_version.fits with the following versions:

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>FM_1_0</td>
<td>TBD</td>
<td>In preparation</td>
</tr>
</tbody>
</table>

5.2 From array coordinates to instrument coordinates

This calfile is defined in full analogy to the corresponding bolometer one. The smaller size and pixel number of the spectrometer arrays will probably mean satisfactory fits with already lower order (first) in u,v (TBC). Again, the conversion is a polynomial of u,v, and chopper angle α.

\[
y = \sum_{i=0}^{N} \sum_{j=0}^{M} \sum_{k=0}^{O} a_{ijk} u^i v^j \alpha^k
\]

\[
z = \sum_{i=0}^{N} \sum_{j=0}^{M} \sum_{k=0}^{O} b_{ijk} u^i v^j \alpha^k
\]
Where the units are:
y,z [arcsec]
u,v [mm]
α [degree]

Calfile:

*With separate entries for both blue and red spectrometer:*

For both y and z:

Maximal orders N,M,O to be considered
(N+1)*(M+1)*(O+1) polynomial coefficients

For ILT, we propose to define a calfile of the same format but with y, z interpreted in mm as y, x offsets of the xy stage for a centered pinhole. ILT shows the red/blue alignment to be pretty good but separate calfile entries are kept.

The derivation of this calfile is based on PCD 4.1.2 and 4.1.3

This calfile is called PacsCal_SpecArrayInstrument_version.fits with the following versions:

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>FM,1,0</td>
<td>TBD</td>
<td>In preparation</td>
</tr>
</tbody>
</table>

### 6 Relation to uplink calibration files

The document “AOT scripts: Functional verification” by Diego Cesarsky (Draft 0.1, 14-Nov-2007) gives, among other things, an overview of the uplink calibration files that are stored in the HCSS (!) CVS repository under develop/data/observingmodes/pacs/calibrationfiles. We give here, without more detail, a reference to the uplink calibration files that are related to spatial calibration. See the headers of the calu files for more detail.

#### 6.1 Uplink calibration files that have to be reflected in IA for proper spatial calibration

<table>
<thead>
<tr>
<th>No</th>
<th>Identification</th>
<th>Comment</th>
<th>IA calfile</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>PACSyzoffsets</td>
<td>Offset to virtual aperture, skew correction</td>
<td>TBD, needed</td>
</tr>
</tbody>
</table>

To ensure proper pointing of the chopped/noded AORs on the source, the RA, DEC of the source specified by the user has to be offset to get the RA, DEC at which the PACS virtual aperture, which assumes chopper at optical zero, has to be pointed. The RA, DEC requested by the user is first offset by the xy offsets and then the SIAM applied to convert into ACA coordinates.

The offset can be specified in the calu file in Z and Y with an additional skew correction to capture misalignments of chopper and spacecraft Y direction.
6.2 Other uplink calibration files related to spatial calibration

<table>
<thead>
<tr>
<th>No</th>
<th>Identification</th>
<th>Comment</th>
<th>IA callfile</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>OBCP3params</td>
<td>Includes chopper ‘dither’ for Phot point source AOR</td>
<td>—</td>
</tr>
<tr>
<td>22</td>
<td>PHOT_CHOP_params</td>
<td>Chopper throw values for HSPOT input</td>
<td>—</td>
</tr>
<tr>
<td>23</td>
<td>PHOT_CHOP_sky</td>
<td>Chopper sky [arcsec] – command calibration table</td>
<td>—</td>
</tr>
<tr>
<td>25</td>
<td>SMALL_SRC_params</td>
<td>Raster parameters for PHOT, SPEC dithering</td>
<td>—</td>
</tr>
<tr>
<td>31</td>
<td>SPEC_CHOP_params</td>
<td>As PHOT_CHOP_params for SPECTRO</td>
<td>—</td>
</tr>
<tr>
<td>32</td>
<td>SPEC_CHOP_sky</td>
<td>AS PHOT_CHOP_sky for SPECTRO</td>
<td>—</td>
</tr>
</tbody>
</table>

These calfiles capture quantities related to chopper and nod throw and to dithers. If defined improperly, they may affect the intended background subtraction strategy and areal coverage of the AOT by improper chopping/nodding, but they will NOT cause incorrect assignment of sky coordinates to a measurement. Nevertheless, they should be defined consistently with the ChopperAngle, PhotArrayInstrument and SpecArrayInstrument downlink calfiles.

7 References

PCD: PACS Calibration Document, PACS-MA-GS-001 Draft 9
IDD: PACS Instrument Description Document Part I, PACS-ME-GR-002 Issue 3.1
The PACS Simulator, PICC-SIM-Saclay-02 Version 1.6.6
Preparing AOTs for the PACS photometer, SAP-PACS-MS-0186-03 Version 2.0
PACS-Chopper Qualification Model Chopper and Control Parameter, PACS-MA-TN-464 Issue 1
PACS OGSE External Point Source Mask Specification, PACS-ME-DS-003 Issue 1
AOT scripts: Functional verification, Diego Cesarsky, Draft 0.1, 14-Nov-2007
PACS OGSE-Sky Distortion, PACS-ME-TN-073 Issue 1

8 Document change record

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Initials</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>2005-06-07</td>
<td>DL</td>
<td>First draft, bololo only</td>
</tr>
<tr>
<td>0.2</td>
<td>2005-06-17</td>
<td>DL</td>
<td>Expanded and included various comments</td>
</tr>
<tr>
<td>0.3</td>
<td>2007-03-08</td>
<td>DL</td>
<td>Small updates, reference to first ILT calfiles</td>
</tr>
<tr>
<td>0.4</td>
<td>2007-10-12</td>
<td>DL,AC</td>
<td>Bolo FM-ILT3 calfiles, first spectrometer draft</td>
</tr>
<tr>
<td>0.5</td>
<td>2007-10-18</td>
<td>DL</td>
<td>Added pixel corners to SubarrayInstrument</td>
</tr>
<tr>
<td>0.6</td>
<td>2007-12-17</td>
<td>DL</td>
<td>CalU, Callfile definitions</td>
</tr>
<tr>
<td>0.7</td>
<td>2008-12-23</td>
<td>DL</td>
<td>Major update</td>
</tr>
<tr>
<td>0.71</td>
<td>2009-03-13</td>
<td>DL</td>
<td>v5 SubArrayInstrument</td>
</tr>
<tr>
<td>0.8</td>
<td>2010-06-01</td>
<td>DL</td>
<td>v5 ArrayInstrument, siam, various</td>
</tr>
<tr>
<td>0.9</td>
<td>2010-07-23</td>
<td>DL</td>
<td>v6 ArrayInstrument</td>
</tr>
<tr>
<td>1.0</td>
<td>2013-06-02</td>
<td>DL</td>
<td>v6 SubArrayInstrument, v7 ArrayInstrument</td>
</tr>
</tbody>
</table>
9 Appendix: Calfile definitions and access methods

The listings below show the access methods of the various calfiles as shown by IA print c.__loc__ (Now no longer possible in this way in the new calibration framework). They should also reflect the order of FITS extensions when reading the calfile into more versatile reduction systems. Be aware of different conventions for the ordering of indices when reading into IA, compared to some popular systems.

9.1 PacsCal_SubarrayArray

The various access methods return 16×32 arrays with u, v, for red and 32×64 arrays for blue. Note that early versions of this calfile do not include the corner coordinates (topleft etc.).

SubarrayArray - Conversion from bolometer subarray coordinates p,q to array coordinates u,v

```java
subarrayArray.getUred() : u values for red array - pixel center
subarrayArray.getVred() : v values for red array - pixel center
subarrayArray.getUblue() : u values for blue array - pixel center
subarrayArray.getVblue() : v values for blue array - pixel center
subarrayArray.getUredtopleft() : u values for red array - top left pixel corner
subarrayArray.getVredtopleft() : v values for red array - top left pixel corner
subarrayArray.getUbluetopleft() : u values for blue array - top left pixel corner
subarrayArray.getVbluetopleft() : v values for blue array - top left pixel corner
subarrayArray.getUredtopright() : u values for red array - top right pixel corner
subarrayArray.getVredtopright() : v values for red array - top right pixel corner
subarrayArray.getUbluetopright() : u values for blue array - top right pixel corner
subarrayArray.getVbluetopright() : v values for blue array - top right pixel corner
subarrayArray.getUredbottomright() : u values for red array - bottom right pixel corner
subarrayArray.getVredbottomright() : v values for red array - bottom right pixel corner
subarrayArray.getUbluebottomright() : u values for blue array - bottom right pixel corner
subarrayArray.getVbluebottomright() : v values for blue array - bottom right pixel corner
subarrayArray.getUredbottomleft() : u values for red array - bottom left pixel corner
subarrayArray.getVredbottomleft() : v values for red array - bottom left pixel corner
subarrayArray.getUbluebottomleft() : u values for blue array - bottom left pixel corner
subarrayArray.getVbluebottomleft() : v values for blue array - bottom left pixel corner
```

9.2 PacsCal_PhotArrayInstrument

The access methods return the maximal orders of the fit polynomials (3 element linear arrays) and the corresponding cubes with coefficients. Note: Separate ‘green’ entries only from v6 on.

PhotArrayInstrument - Conversion from bolometer array coordinates u,v to instrument coordinates y,z

```java
PhotArrayInstrument.getYorderblue() : Maximal polynomial orders N,M,O for y
PhotArrayInstrument.getYcoeffblue() : Cube with coefficients a_ijk for y
PhotArrayInstrument.getZorderblue() : Maximal polynomial orders N,M,O for z
```
PhotArrayInstrument.getZcoeffblue() : Cube with coefficients $b_{ijk}$ for $z$
PhotArrayInstrument.getYordergreen(): Maximal polynomial orders $N,M,O$ for $y$
PhotArrayInstrument.getYcoeffgreen(): Cube with coefficients $a_{ijk}$ for $y$
PhotArrayInstrument.getZordergreen(): Maximal polynomial orders $N,M,O$ for $z$
PhotArrayInstrument.getZcoeffgreen(): Cube with coefficients $b_{ijk}$ for $z$
PhotArrayInstrument.getYorderred() : Maximal polynomial orders $N,M,O$ for $y$
PhotArrayInstrument.getYcoeffred() : Cube with coefficients $a_{ijk}$ for $y$
PhotArrayInstrument.getZorderred() : Maximal polynomial orders $N,M,O$ for $z$
PhotArrayInstrument.getZcoeffred() : Cube with coefficients $b_{ijk}$ for $z$