

**DFMS PDS L2-to-L3  
Data Processing  
Documentation**

**by  
Michael A. Rinaldi  
October 12, 2018  
Version 1.5**

# Introduction

The Rosetta Rosina **D**ouble **F**ocusing **M**ass Spectrometer (**DFMS**) is a high-resolution mass spectrometer (resolution  $m/\Delta m$  3000 at 1% peak height for  $m/z = 28$ ) with a high dynamic range and a good sensitivity.

The DFMS has two operating modes: a gas mode for analyzing cometary gases and an ion mode for measuring cometary ions.

The main parts of the DFMS are the ion source, the analyzer, the detectors, and the zoom optics.

DFMS has three different detectors: MCP/LEDA, CEM, and FC. MCP/LEDA is a combination of an MCP together with a linear detector array with two rows. CEM is a Channel Electron Multiplier that is used for redundancy of the MCP/LEDA and to measure  $m/z$  12. The FC is a Faraday Cup detector and used for absolute calibrations of the instrument. The code focused only on the main detector MCP/LEDA data.

Figure 1 below shows a graphical depiction of the instrument.

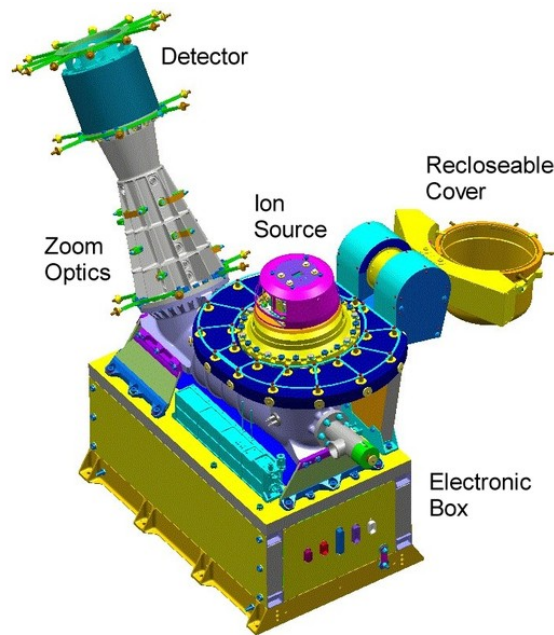


Figure 1. The DFMS Instrument

The instrument has a large number of operational parameters that can be individually adjusted to fit any specific measurement requirements. A specific set of these parameter settings comprises an instrument mode. The instrument is run with specific mode sequences for each mission phase. For more information on DFMS see, *Space Science Reviews*, February 2007, Volume 128, *Issue 1-4*, pp 745-801. Also see, <http://www-personal.umich.edu/~tamas/TIGpapers/2007/Balsiger2007.pdf>

## Scope

This document describes the software written to convert DFMS Planetary Data Systems (PDS) processing level 2 (L2) data to DFMS PDS processing level 3 (L3) data.

The processing level 2 defines data with corrected (edited) telemetry. For this CODMAC level 2 the datasets contain data from all ROSINA sensors (if applicable).

The processing level 3 defines data with physical units. This is the detector current in number of ions vs. mass scale in amu/e. For this CODMAC level 3 the datasets contain Calibrated data and Reduced Data Records foreseen for delivery.

The input (L2) and output (L3) will comply with the PDS archival data product standard. The PDS standard is the de-facto archival standard within the planetary community. Besides NASA, also ESA adopted the PDS standard as the underlying base in building the Planetary Science Archive (PSA), the mission archive for ESA's planetary missions.

The software assumes that L2 PDS data is defined and exists in archived form. This document will describe the process whereby the L2 PDS data is placed into a form that will allow for peak finding, peak fitting, and finally mass scale calibration. The process involves the translation of raw L2 counts data to gain corrected science data. The final data is in the form of ions/spectrum vs mass or the number of ions/spectrum vs mass pixel. The software requires PDS compliant input tables. These are the instrument mode table, overall gain table, pixel gain table, and mass peak search table. These tables are used throughout the L2 to L3 conversion process. In addition to the PDS L3 final data product the software will also produce PDS compliant calibration files called XOFit files.

Following the generation of the DFMS PDS L3 products, a DFMS PDS L3 enhancement software needs to be used to obtain DFMS PDS L3 products with an accurate mass scale. This enhancement software is provided together with the DFMS PDS L2 to L3 conversion software, and is described in SOFT\_L3\_DFMS\_ENHANCEMENT.PDF.

## General Process Description

For each DFMS PDS MCP or CEM L2 data product a respective PDS L3 data product is created, which for MCP will include a calibrated mass scale and corrected signal calibrated to the number of ions for one spectrum. Below there is a section describing the general process to convert a CEM L2 file into its L3 counterpart. This is followed by a section describing the processes converting an L2 MCP file into an L3 file.

The MCP L2 data products are classified into three categories: GCU (Gas Calibration Unit) measurements and two non-GCU measurements. The first nonGCU measurement is identified as SLF (Self Calibrated) and the second as nonSLFnonGCU. SLF measurements are L2 files with commanded masses that may contain known peaks ( $H_2O, CO_2, \dots$ ) which can be used for verification/calibration. nonSLFnonGCU L2 files will contain unknown masses or peaks that couldn't be used for SLF calibration.

GCU measurements are made by mode sequences that are run roughly every month and designed to allow for periodic in-flight calibration from gases with known species. GCU modes will have an advantage in accurate mass-scale calibration as the measured signal represents a known gas mixture with peaks of sufficient amplitude that cover a wide mass range. The SLF modes assume the GCU mass scale and then re-calibrate based on SLF peaks found. The nonGCUonSLF modes are expected to typically produce higher uncertainties since calibration is accomplished using prior linear fit information from both GCU and SLF x0Fit files. The nonGCUonSLF measurements will therefore reference recent GCU and SLF measurements in order to use their mass-scale calibration information.

The general L2 to L3 conversion processes is comprised of two major steps.

**Phase I**, the first step, identifies all GCU and SLF files available in a block of data and processes them to extract tables of commanded mass ( $m_0$ ) versus the mass scale calibration variable (ie., the calibration pixel offset point,  $pix_0$ ). These tables are organized into 22 distinctive sets. For each type of calibration file available (GCU or SLF) there are distinct arrays for low resolution / high resolution (LR/HR), low mass / medium mass / high mass (LM/MM/HM), and finally Row-A / Row-B. Where Row A/B refer to the two DFMS MCP/LEDA detector arrays. Medium mass differentiation is only required for SLF modes. Tables 1 and 2 show the corresponding arrays used to delineate the various types of resolution and masses. These arrays hold  $m_0$  and its corresponding calibration value of  $pix_0$ .

| #  | Array Name | Variable Description                        |
|----|------------|---|
| 1  | gcuLmLrA   | GCU mode Low Mass Low Resolution, Row A     |
| 2  | gcuLmLrB   | GCU mode Low Mass Low Resolution, Row B     |
| 3  | gcuHmLrA   | GCU mode High Mass Low Resolution, Row A    |
| 4  | gcuHmLrB   | GCU mode High Mass Low Resolution, Row B    |
| 5  | gcuLmHrA   | GCU mode Low Mass High Resolution, Row A    |
| 6  | gcuLmHrB   | GCU mode Low Mass High Resolution, Row B    |
| 7  | gcuMmHrA   | GCU mode Medium Mass High Resolution, Row A |
| 8  | gcuMmHrB   | GCU mode Medium Mass High Resolution, Row B |
| 9  | gcuHmHrA   | GCU mode High Mass High Resolution, Row A   |
| 10 | gcuHmHrB   | GCU mode High Mass High Resolution, Row B   |

Table 1. Phase 1 GCU files segregation description

| #  | Array Name | Variable Description                        |
|----|------------|---|
| 11 | slfLmLrA   | SLF mode Low Mass Low Resolution, Row A     |
| 12 | slfLmLrB   | SLF mode Low Mass Low Resolution, Row B     |
| 13 | slfMmLrA   | SLF mode Medium Mass Low Resolution, Row A  |
| 14 | slfMmLrB   | SLF mode Medium Mass Low Resolution, Row B  |
| 15 | slfHmLrA   | SLF mode High Mass Low Resolution, Row A    |
| 16 | slfHmLrB   | SLF mode High Mass Low Resolution, Row B    |
| 17 | slfLmHrA   | SLF mode Low Mass High Resolution, Row A    |
| 18 | slfLmHrB   | SLF mode Low Mass High Resolution, Row B    |
| 19 | slfMmHrA   | SLF mode Medium Mass High Resolution, Row A |
| 20 | slfMmHrB   | SLF mode Medium Mass High Resolution, Row B |
| 21 | slfHmHrA   | SLF mode High Mass High Resolution, Row A   |
| 22 | slfHmHrB   | SLF mode High Mass High Resolution, Row B   |

Table 2. Phase 1 SLF files segregation description

Assuming enough points exist (2 points for GCU and 3 for SLF, *to be set in DFMS\_Constants.dat*) we attempt to create a linear fit to the points,  $m_0$  vs  $pix_0$ , for each of these array types. The resulting linear fit parameters are written to PDS compliant files (x0 fit files), which are later used (Phase II) to calculate the best  $pix_0$  for a given L2 file. The purpose of Phase I is to create the most recent

set of GCU and/or SLF x0 fit files. Each block of L2 files processed will typically create only a subset of the x0 fit files corresponding to the 22 described above. This is so because a block of data is typically associated with a similar mass calibration. The similar mass calibration will in general only sample a subset of the masses and resolutions, depending on the data block processed. For example some mode sequences contain no Low-Resolution (LR) modes at all therefore only High-Resolution (HR) (Table 1, rows 5-10, or, Table 2, rows 17-22) pix0 fit files will be created. This is true for mass sampling as well. Many mode sequences will not contain many or enough High mass (HM) files, for example, to allow the creation of a x0 fit file.

**Phase II**, the second step, runs through each L2 file (GCU, SLF, or nonGCUonSLF) to perform final mass scale calibration using the fitting parameters in the relevant x0 fit files to calculate the best  $pix_0$ . An attempt is also made to characterize the accuracy of the mass scale by performing a part per million deviation ( $PPM_{Dev}$ ) calculation and thus quantify a mass scale quality ID where applicable and then writes final requisite information to the PDS compliant L3 file. The Phase II part of the software uses similar logic to Phase I to find the peak in the L2 data. See the next section for the specific processes involved.

## The CEM L2 to L3 Process Description

With an integration time of typically one second the recording of a whole mass spectrum can be measured with the Channel Electron Multiplier (CEM) detector. The process measures masses from 12 to 140 amu/e.

To evaluate CEM data, the calculation of the ion current for CEM is described and the mass shift between two steps to calculate the mass scale. The CEM detector is mainly operated in digital mode, where CEM operates in a constant high gain plateau at a front voltage of -2337V. The repetition voltage in the CEM is -100V. In this mode events are counted, more precisely the arrival of a secondary electron cloud, caused by an impacting ion. The saturation for counting is around  $2 \times 10^6$  events per second. A spectrum of the CEM detector is recorded by shifting the ion beam in little mass steps over the entrance slit of the CEM detector. The nominal integration time per step is 1000 ms and the number of steps is mass and resolution dependent.

|                 | Step width                   | Number of steps                 |
|-----------------|------------------------------|---------------------------------|
| Low Resolution  | $\Delta m = \frac{m}{1000}$  | $\sim \frac{140}{\sqrt{m}} + 1$ |
| High Resolution | $\Delta m = \frac{m}{10000}$ | $\sim \frac{240}{\sqrt{m}} + 1$ |

Table 3. CEM mass step width and number of steps per spectrum

The shifting of the ion beam over the entrance of the CEM detector causes an overlap of the ion beam. The overlap is calculated as,

$$C_{ol} = \frac{D \cdot \Delta m}{W_s \cdot m}$$

Where D is the dispersion (127,000  $\mu\text{m}$ ),  $\frac{\Delta m}{m}$  is the step width and  $W_s$  is the width of the entrance slit of 25  $\mu\text{m}$ . For LR the overlap correction factor  $C_{ol} = 5.08$  and for HR  $C_{ol} = 0.508$ . The ion current of the species ( $I_{CEM,i}$ ) of the CEM detector operated in digital mode is calculated as,

$$I_{CEM,i} = \frac{C_{ol} \cdot \text{Sum}_i}{t}$$

The  $\text{Sum}_i$  is the sum of the peak of species I and t is the integration time per step (nominal 1 s).

The Process to convert one Level 2 PDS CEM file into a Level 3 is fairly straight forward.

1.) We first find the resolution per step ( $\frac{\Delta m}{m}$ ) based on the mode resolution this is,

$$\frac{\Delta m}{m} = 1/1000 \quad \text{for LR}$$

$$\frac{\Delta m}{m} = 1/10000 \quad \text{for HR.}$$

2.) Next we calculate the mass dependent step value (using the commanded mass  $m_0$ ),

For HR this is calculated as,

$$\begin{aligned} \text{step0} &= 3.6713*m_0 - 10.85 & m_0 \geq 12 \text{ and } m_0 \leq 15: \\ \text{step0} &= 3.4728*m_0 - 31.38 & m_0 \geq 16 \text{ and } m_0 \leq 18: \\ \text{step0} &= -26.49*\log(m_0) + 106.0 & m_0 \geq 19 \text{ and } m_0 \leq 45 \\ \text{step0} &= 6.0 & m_0 \geq 46 \text{ and } m_0 \leq 140: \end{aligned}$$

For LR this is calculated as,

$$\text{step0} = -8.10*\log(m_0) + 40.68 \quad \text{for all masses}$$

3.) Next we calculate the CEM signal as,

$$\text{signal} = D*\frac{\Delta m}{m} / (W_s * t)$$

4.) Next we calculate the mass at step(i) as,

$$m_i = m_0*((\text{step}_i - \text{step}_0)*\frac{\Delta m}{m} + 1.0)$$

5.) Finally the current at i is calculated as,

$$I_i = \text{signal} * \text{Sum}_i$$

The final results written to the L3 file in PDS format.



## The MCP/LEDA L2 to L3 Process Description

The method used to convert one L2 file to one L3 file consists of several steps. The first is to ascertain the offset correction factor. Next the L2 pixel data is corrected for Overall Gain and individual Pixel Gain. Once the data has been corrected for Offset and Gain we can then proceed to calculate the number of ions attributed to each pixel. The final step is to perform the mass calibration. The calibration of course depends on the L2 file mass and mode.

### The Offset

The LEDA offset correction is calculated for every L2 file. This correction is modeled by a 3<sup>rd</sup> order polynomial fit to the offset data. The polynomial is of the form,

$$\text{overallOffset} = c_3x^3 + c_2x^2 + c_1x + c_0 \quad (1)$$

where x is the pixel number (1-512).

Each L2 file is fit with a 3<sup>rd</sup> order polynomial equation of the form in Eqn. (1) above. The polynomial coefficients are,  $c_3, c_2, c_1, c_0$ . In order to create a smooth data set that best represents the best offset running from pixel 1-512 we remove the first and last 20 pixels and only fit pixels 20-492. This is done to remove the highly variable edge pixels that would tend to skew the true offset. We also remove the peak(s) from the data. The process of removing peaks is accomplished by reading input data consisting of main peak(s) location and pixel start/end data. This peak exclusion data is read from two input files currently named, DFMS\_20140401-20160127\_Peak\_Exclusion.dat, and DFMS\_20160127-20161001\_Peak\_Exclusion.dat. These two files consist of up to ten peak boundaries for each relevant mass. The peak exclusion function checks if each L2 file contains the relevant mass and then performs the peak subtraction, for each peak expected to be in the L2 file data. Excluding peaks is done to present a more representative offset without the overweighting of the peaks. Thus when performing the 3<sup>rd</sup> order polynomial fit to the offset data we can start with a set of data that contains no edge effects or peaks. For each L2 file we also write the calculated polynomial coefficients to the housekeeping portion of the L3 Housekeeping object (for specific names see Note).

The *overallOffset* is then subtracted from the original L2 raw data counts.

$$\text{offsetCorrectedData} = \text{rawData} - \text{overallOffset} \quad (2)$$

A typical LEDA offset plot and calculated polynomial fit is shown in Figure 3,

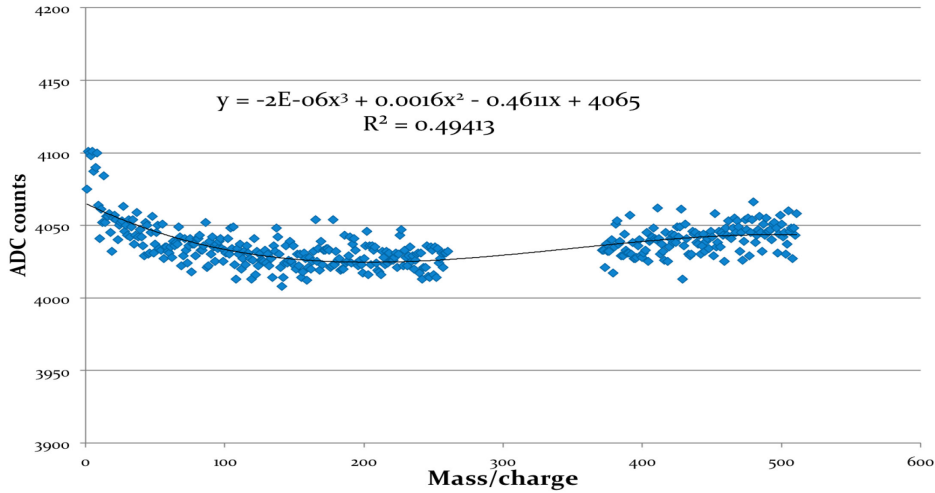


Figure 3. A plot of ADC (Raw) counts vs pixel number (mass/charge)

**Note:** The coefficients  $c_0$ ,  $c_1$ ,  $c_2$ , and  $c_3$  are written to the housekeeping section of the output L3 file. They are named ROSINA\_DFMS\_SCI\_OFF\_COEFF\_C<sub>x</sub>\_y, where  $x = 1, 2, \text{ or } 3$  and  $y = A \text{ or } B$ .  $c_0$  is written as well and is called the ROSINA\_DFMS\_SCI\_OFF\_LEVEL\_y, where  $y = A \text{ or } B$ .

### Overall Gain and Pixel Gain Correction

After first subtracting the offset as shown above we then apply the overall gain and then pixel gain corrections (Eqs.3,4). The time-dependent overall gain correction (e.g. tables/Gain/FS/GAIN\_TABLE\_YYYYMMDD\_FS.TAB) is a mode dependent value that represents the average number of electrons that each ion creates when it impinges the DFMS MCP sensor. Since the raw counts are based on the number of electrons detected on the LEDA and not the actual number of ions this signal ‘amplification’ has to be corrected for by dividing the overall gain correction from the offset corrected raw data. This results in the gain corrected data,

$$gainCorrectedData = offsetCorrectedData / overallGain \quad (3)$$

The overall gain is somewhat time dependent therefore the code will look for the gain tables nearest the L2 file data and linearly interpolate or extrapolate to find the best overall gain. Next we apply the individual pixel gain correction. The pixel gain (e.g. tables/Gain/FS/20140725/PIXGAIN\_20140725\_M\_FS\_GS16.TAB) corresponds to a correction for each pixel in the MCP sensor field. It is time varying and is often re-measured to compensate for MCP sensor degradation (ageing). Thus for each pixel  $i$ , in the MCP we correct for pixel gain as,

$$pixelCorrectedData[i] = gainCorrectedData / pixelGain[i] \quad (4)$$

An example of a pixel gain spectra is shown in Figure 4:

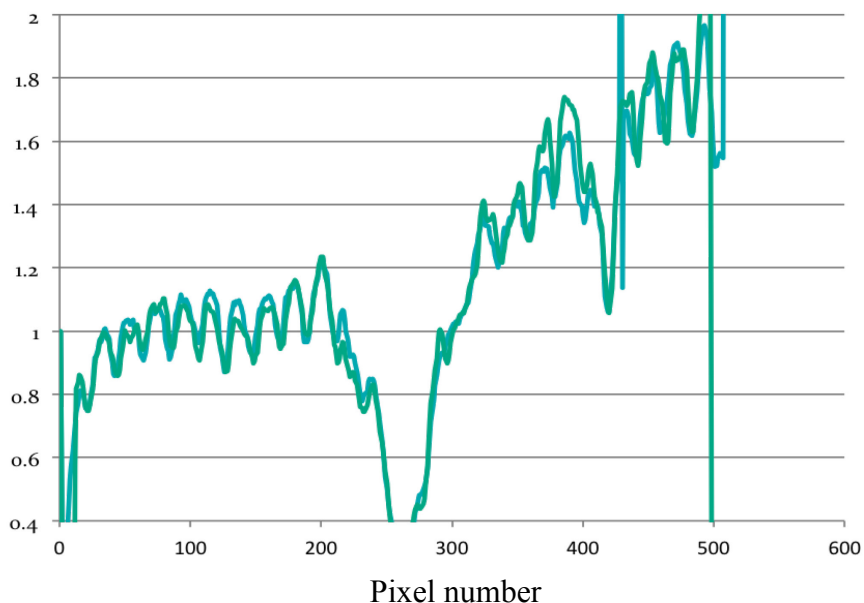
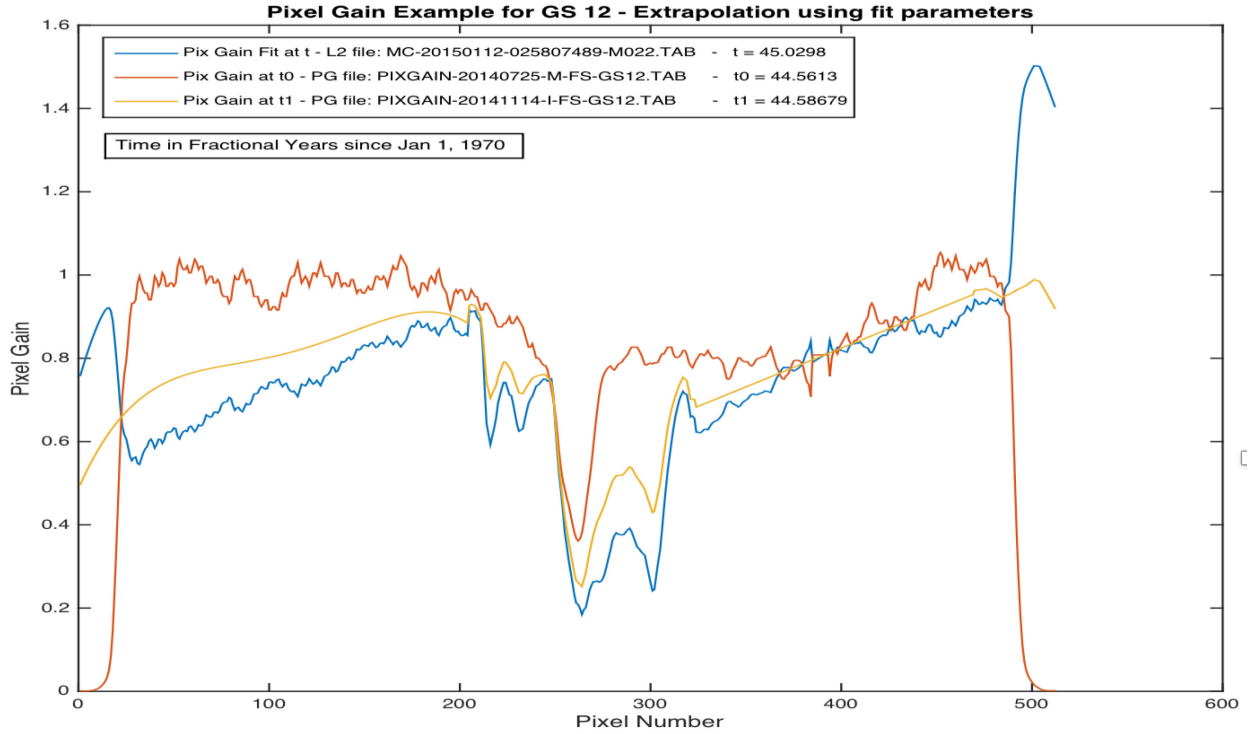


Figure 4. A representative pixel gain spectra

With the availability of new pixel gain tables the code has been modified to include pixel gain versus time interpolation. In many cases, where 2 or more pixel gain files exist with the same gain step over a time range, this will allow for pixel gain values to be computed from linear interpolation within an L2 file date range. This proceeds as follows. Before any L2 to L3 conversion is done the code loads all existing pixel gain tables and sorts them according gain step value. For each gain step value where 2 or more files exist we perform a linear interpolation between pixel gain value (at each pixel number) versus time. The resulting interpolation parameters are stored in global arrays and later used to calculate the proper pixel gain value for the specific L2 file date. In situations where only one pixel gain file exists for a specific L2 gain step the code will look for the pixel gain table nearest in time. In other possible situations where the L2 gain step is not represented by any pixel gain table the code will look for the pixel gain table nearest in time and then nearest in gain step value. Figure 5 below shows a typical Interpolation result using the technique described above



**Figure 5. Example of interpolation used to find more accurate Pixel Gain**  
**Calculating the Total Number of Ions**

The final calculation is to convert the *pixelGainCorrectedData* to the physically meaningful value of the number of ions per spectrum. This is done on a per pixel (*i*: 1-512) basis simply as,

$$Ions[i] = pixelGainCorrectedData[i] * C_{ADC} * C_{LEDA} / (Q * y_s) \quad (5)$$

Where,

$$C_{ADC} = 6.105 \times 10^{-4} \text{ V}$$

$$C_{LEDA} = 4.22 \times 10^{12} \text{ F}$$

$$Q = 1.602 \times 10^{-19} \text{ C}$$

$$y_s = 1.0 \text{ (detector yield per species normalized to N}_2 \text{ at 3000V)}$$

The number of ions per spectrum is the final result of the L2 data conversion. Along with the mass scale, which is calculated by finding the correct *pix0* calibration value, the number of ions per spectrum is written to the L3 file. The ROSINA\_DFMS\_SCI\_SIGNAL\_CAL\_VAL\_A and ROSINA\_DFMS\_SCI\_SIGNAL\_CAL\_DEV\_A for row A and accordingly for row A can be found in the L3 output.

Due to a mass dependence on yield the corrected Ions per spectrum is calculated as follows.

For ion mass < 70 the yield correction is defined as,

$$y_{Corr} = 1.0 / (4.4892 \times 10^{-7} * m^4 - 8.8158 \times 10^{-5} * m^3 + 6.4995 \times 10^{-3} * m^2 - 0.2223 * m + 3.4922) \quad (6)$$

For ion mass  $\geq 70$  the yield correction depends on resolution as follows.

For Low Resolution spectra the yield correction is defined as,

$$yCorr = 1.0 / (-2.400438 * 10^{-3} * m + 0.5684252) + 0.8 \quad (7)$$

For High Resolution spectra the yield correction is defined as,

$$yCorr = 1.0 / (-2.400438 * 10^{-3} * m + 0.5684252) \quad (8)$$

In (Eq. 6-7) above 'm' is the DFMS detector commanded mass.

Therefore, the yield corrected Ions per spectrum is,

$$Ions[i] = yCorr * pixelGainCorrectedData[i] * C_{ADC} * C_{LEDA} / (Q * y_s) \quad (9)$$

The calculation of Ions per spectrum of (Eq. 9) replaces that of (Eq. 5).

## Peak Finding

Once we have established the ion number per pixel we proceed to identify all the peaks present in the data and the main peak of interest, which we will fit later. A general peak finding algorithm is applied to the raw data. This algorithm finds all peaks above a predefined threshold. The threshold is defined as mean offset ( $c_0$ ) calculated above plus  $\sim 5$  times the standard deviation to this mean offset. The actual number of std deviations used is set by the user in the input configuration files DFMS\_Constants.dat Given a possible set of peaks (1 to N) finding the right peak for fitting involves two criteria. The first is specifying the pixel boundaries within which to look and secondly looking for the tallest peak within that boundary. For GCU files the peak pixel boundary is specified in the MPS table. For the remainder of the L2 files (nonGCU) the peak is assumed to exist within the pixel range 20 – 492. The detector edge pixels are left out since they have various pixel calibration issues that distort the data. There is an ambiguity in that for some commanded masses the tallest peak is not the correct peak to fit. This happens in situations (eg., commanded mass 36) where the tallest peak within the 20-492 pixel boundary may not be the peak of interest. Such tall peaks are typically near the edge of the accepted boundary (eg., pixel 430). In this case the code applies an algorithm where the pixel boundary is set to 210-300. Within this smaller pixel boundary we search for any peaks that are at least 50% of the height of the tallest peak in the full boundary region (20-492). If one or more are found then the tallest of these is chosen as the main peak used by the fitting algorithm.

## Peak Fitting

With the main peak identified the next step is to fit a simple Gaussian to it. The initial peak search algorithm supplies the needed initial values of peak location, height and width. These initial values are used as initial conditions for the non-linear least squares fit. Specifically, the fitting algorithm performs a fit of a 2D (X,Y) data set to a simple Gaussian function by adjusting the initial set of parameter values using a Levenberg-Marquardt Algorithm (LMA). The routine iterates until an improvement of the chi-squared statistic is achieved from the initial parameters. Once the change in the chi-squared reaches a predefined value (eg., 1.0e-8) we assume the fit has converged to the best parameter values. The last fit coefficients are then used as the accepted fit parameters. See Figure 6 for an example of the resulting Gaussian fit. Since we are mainly interested in the location of the peak we can use a simple Gaussian function. In actuality a Pseudo-Voigt Gaussian is a better function since it will generally do a better job at fitting the wings of the peak but using this type of function introduces longer compute times and more fitting parameters. Nevertheless it maybe considered for future implementation if other parameters like the precise peak fit area are required.

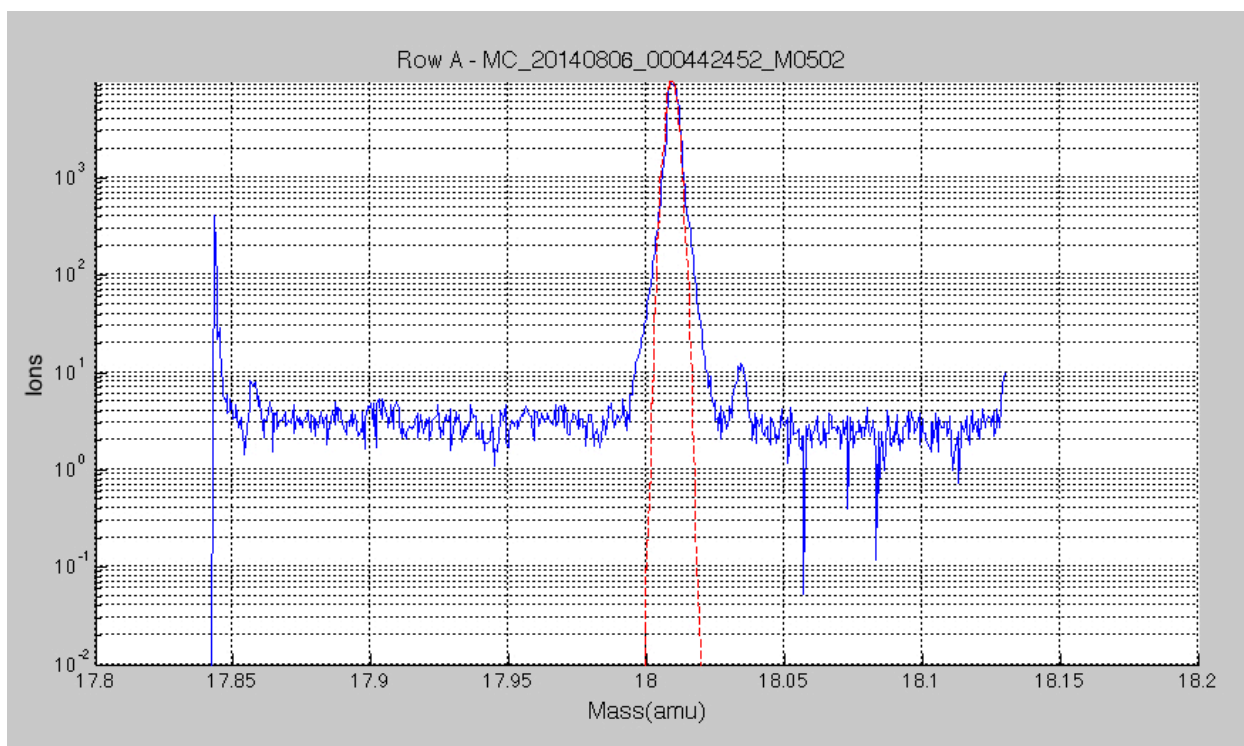


Figure 6. An example of a Gaussian fit to the Ion Number vs Mass

## Mass Scale Calibration

The purpose of mass scale calibration is to calculate the correct value of  $pix_0$  (the calibration pixel offset point). This is required in order to find the best possible mass scale. The calibration requires that we make optimal use of the known mass peaks that exist. The most direct and accurate calibration comes from data collected during GCU modes where known gases are used to calibrate the instrument. Since GCU modes are not run very often we can also make use of peaks from known mass species near the spacecraft. These are collectively called SLF masses since we attempt to use them as SeLF-calibration peaks. These commanded masses are shown in Table 4 below.

| Commanded Mass | Expected Species              |
|----------------|-------------------------------|
| 13             | CH                            |
| 15             | CH <sub>3</sub>               |
| 15.01          | CH <sub>3</sub>               |
| 16             | <sup>16</sup> O               |
| 16.51          | <sup>16</sup> O               |
| 18             | H <sub>2</sub> O              |
| 18.16          | H <sub>2</sub> O              |
| 21.98          | CO <sup>2++</sup>             |
| 22             | CO <sup>2++</sup>             |
| 24             | C <sub>2</sub>                |
| 24.18          | C <sub>2</sub>                |
| 25             | C <sub>2</sub> H              |
| 35.4           | C <sub>3</sub> H              |
| 37             | C <sub>3</sub> H              |
| 38.94          | C <sub>3</sub> H <sub>3</sub> |
| 39             | C <sub>3</sub> H <sub>3</sub> |
| 42.83          | CO <sub>2</sub>               |
| 44             | CO <sub>2</sub>               |
| 47.12          | SO                            |
| 48             | SO                            |
| 50             | C <sub>4</sub> H <sub>2</sub> |
| 51             | C <sub>4</sub> H <sub>3</sub> |
| 51.83          | C <sub>4</sub> H <sub>3</sub> |
| 53             | C <sub>5</sub>                |
| 60             | OCS                           |
| 62.72          | OCS                           |
| 75.89          | C <sub>6</sub> H <sub>6</sub> |
| 76             | CS <sub>2</sub>               |
| 78             | C <sub>6</sub> H <sub>6</sub> |
| 91             | C <sub>7</sub> H <sub>7</sub> |
| 91.83          | C <sub>7</sub> H <sub>7</sub> |

Table 4. SLF calibration masses



[**A Note on modifying the SLF masses**] *The values used in Table 4 above are variable in that they depend on the expected species that will be found at various stages of the Rosetta mission. The user may change the commanded masses that the code uses for SLF calibration. The number of masses used is set by the variable NUMSLFPKMASSES which is defined in the source code DFMS\_definedConstants.hh. The associated SLF commanded mass values are set in the code DFMS\_process\_L2\_to\_L3.cc. Specifically, the code that defines the currently used masses is,*

*double* slfMasses[NUMSLFPKMASSES] = {13, 15, 15.01, 16, 16.51, 18, 8.16, 21.98, 22, 24.0, 24.18, 25, 35.4, 37, 38.94, 39, 42.83, 44, 47.12, 48, 50, 51, 51.83, 53, 60, 62.72, 75.89, 76, 78.0, 91, 91.83};

*The user must modify this array to add or remove SLF masses and then adjust the value of the variable NUMSLFPKMASSES (in DFMS\_definedConstants.hh) to reflect the current number of SLF masses used.*

The mass scale calibration process proceeds in two phases. The first phase creates the latest  $pix_0$  vs  $m_0$  linear fit (Eq. 10) and stores the offset ( $a$ ) and slope ( $b$ ) parameters in a PDS compliant file (or, x0Fit file). The set of x0 fit files define a time series of fit parameters. In Phase II we use the latest appropriate x0 fit file parameters to calculate  $pix_0$ .

Phase I – During this phase we collect all L2 files that are GCU modes and SLF files whose commanded masses and resolutions are described in Table 1 and 2. The files are first ordered by type, GCU modes are processed first followed by SLF. The resultant list of GCU/SLF files is then time ordered. With an ordered set of L2 files we then perform peak finding and peak fitting to find  $pix_0$  for each  $m_0$ . For each type of file (GCU or SLF) we store the  $pix_0$ ,  $m_0$  data into array based on mass and resolution (Table 1,2). Once all the data is collected we proceed to perform linear fitting to each set of  $pix_0$  vs  $m_0$  set of data. The fit is of the form,

$$pix_0fit = a + b * m_0 \quad (10)$$

The resulting fit parameters  $a, b$  as well as the uncertainty (Eq. 18) in the final  $pix_0fit$  value are written to files with the naming convention x0\_TYP\_YYYYMMDD\_HHMMSS\_KIND.TAB. Where *TYP* is one of GCU or SLF, *YYYYMMDD\_HHMMSS* is the time stamp (based on the time of the first L2 file -  $pix_0, m_0$  pair used) , and *KIND* is one of LMLR (Low Mass Low Resolution), and similarly HMLR, LMHR, HMHR. Thus ideally there should exist 8 distinct x0 fit files. There must be at least 4  $pix_0, m_0$  pairs to perform a fit of GCU type and 3  $pix_0, m_0$  pairs for SLF type. The resulting fit files will be added to a directory of existing fit files with the result that the appropriate x0 fit file closest in time to the L2 file being processed will be used for final mass scale calibration.

Phase II - This phase follows a similar set of procedures (codes) as Phase I. We again perform peak finding and fitting to gain the pixel location of the mass peak. We then attempt to find the best mass scale as follows. We assume we have created or have access to the latest x0 fit files. The calibration is performed differently for the different types of L2 file types. For GCU files we have, as given, the specific commanded mass and the corresponding known value for this mass. The calculation of  $pix_0$  for GCU is simply given by,

$$pix_0 = x - \log\left(\frac{m}{m_0}\right)/C \quad (11)$$

where,  $x$  is the pixel location of the mass peak,  $m_0$  is the commanded mass,  $m$  is the known mass from the Mass Peak Search Table, and,

$$C = \frac{25.0}{(DISP * zoom)} \quad (12)$$

*zoom* is 1.0 for low resolution modes and 6.4 for high resolution modes.

In Eq. (12) the dispersion, DISP, is constant for masses < 70. For masses >= 70 the dispersion follows a power law. In general, DISP, is calculated as follows,

$$DISP = 127000, \quad \text{for } m_0 < 70 \quad (12a)$$

$$DISP = 382200 * m_0^{(-0.34)}, \quad \text{for } m_0 \geq 70 \quad (12b)$$

For the case of SLF files the process is different. Recall that an SLF file is defined as an L2 file where the commanded mass ( $m_0$ ) is such that it is likely that a known mass peak lies somewhere between pixels 1-512. Therefore the calibration process proceeds in two steps. Using the commanded mass  $m_0$ , we first calculate an initial  $pix_0$  given by the latest GCU calibration x0 fit parameters. Namely,

$$pix_{GCU0} = a_{GCU} + b_{GCU} * m_0 \quad (13)$$

where,  $a_{GCU}$  is the offset and  $b_{GCU}$  is the slope (see Eqn. 10) of the linear fit.

Using  $pix_{GCU0}$  we calculate a mass scale as,

$$m(i) = m_0 * e^{(C * (x(i) - pix_{GCU0}))} \quad (14)$$

This mass scale is then searched for the known SLF masses (one of the masses in Table 4). If one of these masses lies within the bounds of this mass scale we then check if the known peak location lies within +/- 0.1amu of the calculated  $pix_0$  for each endpoint. That is, if  $pix_1$  corresponds to  $m+0.1$ , and  $pix_2$  corresponds to  $m-0.1$  (calculated from Eq. (10)) then if the peak location  $x$  is within the bounds of  $pix_1$  and  $pix_2$  we assume that the peak represents the known SLF mass. We then calculate the SLF  $pix_0$  calibration values as,

$$pix_{SLF0} = a_{SLF} + b_{GCU} * m_0 \quad (15)$$

where  $a_{SLF}$  is the offset (a) linear fit parameter from the latest Phase I SLF x0 fit file and  $b_{GCU}$  is the slope (b) linear fit parameter from the latest Phase I GCU x0Fit file. This calculation assumes that the slope is best represented by the GCU fit while the offset by the SLF fit.

With the best  $pix_0$  being established by either Eq (13) for GCU files or Eq (15) for SLF files we then proceed to calculate the best mass scale using Eq. (14).

**[Note on DFMS GCU failure]** *As of Dec. 28, 2014 the GCU calibration unit is no longer functioning. Thus we define two different calibration scenarios based on the L2 data time stamp. For L2 files generated before Jan 3, 2015 we can use the latest GCU and SLF  $pix_0$  fit files as described above. For all L2 data after Jan 3, 2015 we now modify Eq 15 above so that we use the SLF slope instead of the GCU slope. Thus for all L2 data after Jan 3, 2015 (Eq 15) is replaced with (Eq 15a) as,*

$$pix_{SLF0} = a_{SLF} + b_{SLF} * m_0 \quad (15a)$$

Lastly, for nonGCUnonSLF files, we use the latest x0 fit files for both GCU and SLF to calculate  $pix_0$  as,

$$pix_0 = a_{SLF} + b_{GCU} * m_0 \quad (16)$$

Once again for all L2 data after Jan 3, 2015 (Eq 12) is replaced with (Eq 12a) as,

$$pix_0 = a_{SLF} + b_{SLF} * m_0 \quad (16a)$$

Therefore Eq. (14) calculates the best mass scale for GCU files while Eqs (15 or 15a) and (16 or 16a) calculate the best mass scale for SLF and nonGCUnonSLF files respectively using Eqs. 15,16 for time periods before Jan 3, 2015 and Eqs. 15a,16a for time periods after Jan 3 2015.

ROSINA\_DFMS\_SCI\_xxx\_PIXEL0\_y and ROSINA\_DFMS\_SCI\_xxx\_PIXEL0\_UNC\_y provide the information of the px0 used to calibrate the mass scale provided in the L3 file, while xxx can be GCU or SLF and y is either A or B. If only GCU calibration was used the according PIXLE0 for SLF is N/A.

### The Uncertainty in pix0

The pix0 uncertainty is dependent on resolution. Due to mass scale calibration complexities with high-resolution modes the pix0 uncertainty for all high-resolution modes is fixed at 20.0 pixels. This software does not attempt to use any special processing for high-resolution modes. This software uses the same calibration technique for both low and high resolution modes. Therefore the fixed high resolution pix0 uncertainty. For low-resolution modes the uncertainty is calculated based on the data spread that is used to form the x0 fit file. For every fit file there is an uncertainty in the pix0 value that is calculated as,

$$\sigma_{pix0} = \sqrt{\sum_{i=1}^N \frac{(pix0Fit[i]-pix0[i])^2}{N-2}} \quad (17)$$

where N is the number of  $m_0, pix_0$  pairs used to calculate the fit. For self calibrated SLF files the total error in pix0 is calculated as,

$$\sigma_{tot} = \sqrt{\sigma_{pix0}^2 + 5.0^2} \quad (18)$$

This includes a minimum error of 5 pixels in addition to the error calculated from the fit.

## Establishing the Mass Scale Quality

Once the mass scale has been calculated its accuracy or quality needs to be established. This mass scale quality depends on comparison of the peak mass derived from the mass scale to the known mass. We attempt to define a parts per million difference ( $PPM_{Dev}$ ) between the known and mass scale derived peak mass. For both GCU and SLF files this is simply calculated as,

$$PPM_{Dev} = \left| \frac{m_{known} - m(x)}{m(x)} \right| * 10^6 \quad (19)$$

Where  $m(x)$  is the mass at peak location  $x$  and  $m_{known}$  is the known mass of that peak. If this  $PPM_{Dev}$  is  $< 500$  we conclude that the mass scale is of good quality and the fit quality ID is set to 0.

For nonGCUonSLF files the process is different. Since we don't have a known peak mass the best we can do is to infer the  $PPM_{Dev}$  from the best mass scale derived in the latest SLF type  $PPM_{Dev}$  calculations. In this regard we keep track of the  $PPM_{Dev}$  calculated during the processing of the SLF files. Specifically, for each type of SLF (Low/High mass and Low/High resolution and Row A/B) we keep track of the  $PPM_{Dev}$ . When processing a similar type nonGCUonSLF file we simply assign the  $PPM_{Dev}$ , closest in time, of the same type (eg.,  $PPM_{Dev\_LmHrRowA, \dots}$ ). A record of the  $PPM_{Dev}$  calculated for each SLF mass is held in volatile memory during a specific code run. A code run typically covers periods of no longer than a few days worth of data therefore the number of SLF masses and L2 modes is limited and may exclude certain resolution or mass range regions. Often we do not have a candidate for every type of  $PPM_{Dev}$  these then are assigned a value of N/A (Unknown or not Applicable).

ROSINA\_DFMS\_SCI\_AVG\_PPM\_DEV\_y for y A or B provides the parts per million described above for the according file in the L3 output.

## The Quality ID

Lastly quality ID flag for every L2 to L3 conversion is assigned. The Table below defines the various Quality ID flags allowed.

| Quality ID | Description  |
|------------|--|
| 0          | Nominal quality, <u>avg. PPM deviance</u> $< 500$                      |
| 1          | Self-calibrated, GCU <u>avg. PPM deviance</u> $\geq 500$ , SLF $< 500$ |
| 2          | Adopted mass scale <u>avg. PPM deviance</u> $\geq 500$                 |
| 3          | Enhanced Noise   |
| 4          | Not enough peaks found for accurate calibration/verification           |

Table 5. Quality ID definition

## Input Tables

The DFMS code requires several input tables to perform its functions. These are, the Overall Gain table which lists the gain values for the 16 gain steps. There are up to 16 pixel gain tables corresponding to each gain step. There are the Mass Peak Search Tables for GCU and SLF modes. Finally there is the Mode ID Table.

1. Overall Gain Table – This table consists of two possible types, namely the FM (Flight Model on ground) or FS (Flight model in Space) Gain table. Each consists of 16 gain values corresponding to 16 possible gain steps. (See DFMS\_GAIN\_TABLE.FMT for a description of the data fields).
2. Pixel Gain Tables – For each type of Overall Gain Table we have an associated (FM or FS) Pixel Gain table. The pixel gain table is time stamped since the gain values for each pixel are often recalculated do to the deterioration of the MCP detector. If 2 or more pixel gain tables exist for the same gain step then the code will use the interpolation of pixel gain values closest in time. If 1 or less pixel gain tables exist for the same gain step then the code uses the pixel gain table closest in time and then gain step to the L2 file. (See DFMS\_PIXEL\_GAIN\_TABLE.FMT for a description of the data fields).
3. Mass Peak Search Table – These tables are used to associate a known mass value to GCU or SLF commanded mass. They are also used to specify the pixel range over which to look for the known peak. There are two types of MPS table, namely the GCU and SLF. (See DFMS\_MASS\_PEAK\_SEARCH\_TABLE.FMT for a description of the data fields).
4. Mode ID Table – This table lists in ascending order the mode numbers that are defined by the Rosina/DFMS instrument. Each mode number is unique and contains the necessary information used by the code so that the proper mode related settings are defined. Over the life of the Rosina/DFMS experiment new modes are often added to the Mode ID table. As this happens a new Mode ID table is created and time stamped accordingly. The code requires the user to define the specific Mode ID table to be used. This is done in the ConfigFile.dat input file. (See DFMS\_MODE\_ID\_TABLE.FMT for a description of the data fields).

## Code Installation and Environment setup

The code is installed in a user-defined location. This installation directory is then called the DFMS installation directory and defined by the user or system environment variable `DFMS_INSTALL_DIR`. It is the users responsibility to define this environment variable after the installation. On a Unix system this is simply done as,

```
unix> export DFMS_INSTALL_DIR=<install directory>
```

Where *<install directory>* is the location where the `src/` directory exists.

In addition to the source code the installation requires the existence of the GNU open source GSL libraries. The external codes in this library are used for the purpose of performing polynomial fits. In particular the code uses the file `gsl_multifit`. The GNU GSL library is available from <https://www.gnu.org/software/gsl>.

For proper installation, we need to specify the location of the include files,

For Cygwin/Linux setup add packages `libgsl-devel-2.3-2` and `gsl 2.3-2` (or latest available). Once GSL is installed then make sure that the line in the Makefile,

```
CFLAGS += -w -g -Wall -c -I/usr/include/gsl -I/usr/include
```

points to the correct location where the `gsl` include files are installed. The code needs to find the include files (`.h` files associated with `gsl`).

The sharable object files, `libgsl.so.0`, and `libgslcblas.so.0`, must also exist in the default location, namely, `/usr/lib`.

## Building the Executable

The installation source files include the file `Makefile`. This file is used to build the final executable from the source files in the `src/` directory. To build the executable issue the following commands on the unix command line.

```
unix> make clean
```

```
unix> make
```

*make clean* will remove all existing object files in the directory `obj/` as well as the current executable file in the directory `bin/`.

*make* will create new versions of the object files and place them in the `obj/` directory and then link these objects and any libraries into an executable (named *DFMS\_PDS\_L2\_to\_L3*) that will be placed in the `bin/` directory.

## The Configuration Files

The DFMS L2 to L3 processing code requires the user to supply and maintain 5 configuration files. They are *ConfigFile.dat*, *DFMS\_Constants.dat*, *DFMS\_Exclusion\_Times.dat*, *DFMS\_20140401-20160127\_Peak\_Exclusion.dat*, and *DFMS\_20160127-20161001\_Peak\_Exclusion.dat*. Each file is self-documented therefore please inspect the file for the definitions of the required constants. In general the *ConfigFile.dat* configuration file contains non-DFMS specific variable definitions, the *DFMS\_Constants.dat* configuration file contains variable definitions pertaining to the DFMS experiment. Changes in this file will have an impact on the results. The *DFMS\_Exclusion\_Times.dat* contains starting and ending times within which the L2 to L3 processing should be skipped. During these time ranges instrument parameters were changed, therefore a scientific calibration of the data is not feasible. The peak exclusion files are needed in identifying peaks and peak boundaries for a given mass. The peaks will then be removed from offset polynomial fitting input data.

## Initial starting condition of the software

The software needs a set of GCU X0 fit files as starting point. Since the number of GCU calibration measurements is limited this set is limited too. The GCU X0 fit files are produced with the software by processing all available GCU measurements. (These files are located in the folder data/X0fit)

## Array Size and Other Required Values

The file *DFMS\_definedConstants.hh* contains a list of C pre-processor (#define) definitions that must be set in order for the code to compile and execute. These are mainly array size parameters for arrays that cannot be dynamically sized as well as other non DFMS related constants that the code requires.

## Running the Executable

The DFMS L2 to L3 conversion code (*DFMS\_PDS\_L2\_to\_L3*) can be executed in two general ways from the Unix command line.

1. The most common way is to process all L2 files in a given directory. The directory is defined in *ConfigFile.dat* by the variable *sL2sourcePath*. Note that this directory as well as all other directories listed in this configuration file is specified to be relative to the Installation directory *\$DFMS\_INSTALL\_DIR*. With *\$DFMS\_INSTALL\_DIR* properly set then to execute the code simply issue the command,

```
unix> bin/DFMS_PDS_L2_to_L3
```

This will process all L2 files in the directory *sL2sourcePath* and place the resulting L3 files in the directory defined by *sL3outputPath*. The code will generate a time tagged log file with processing information and place it in the directory defined by *sLogPath*. The log files are named,

*DFMS\_processLogFile-yyyymmdd\_hhmmss.log*. The user can optionally set the code (via setting the configuration variable L3INFOTOFILE to 1) to write an easily readable L2->L3 processing summary file into the directory defined by *sL3InfoLogPath*. These files are called, *DFMS\_L3InfoLogFile-yyyymmdd\_hhmmss.log*. The user may also optionally set the code (via setting the configuration variable QLINFOFILE to 1) to write a quick look summary file into the directory defined by *sQLLogPath*. These files are called, *DFMS\_QuickLookLogFile-yyyymmdd\_hhmmss.log* and consist of one line per L2 converted file. Lastly, the user may set the code to create an ascii file containing a table for each L3 file listing the resulting pixel/mass scale/ion number for later plotting using user developed code. Setting the configuration variable DIAGTOFILE to 1 does this. The files created are named, *MC\_<yyyymmdd\_hhmmss>\_M<xxxx>\_Mass\_<mm>\_plot.dat*.

The user can also take advantage of several Command Line Arguments (CLA) to execute the code in specific ways. These are, for example,

1. *unix> bin/DFMS\_PDS\_L2\_to\_L3 -f MC\_20150103\_000337564\_M0600.TAB*. The code will convert only the single file specified after the *-f* argument.
2. *unix> bin/DFMS\_PDS\_L2\_to\_L3 -l* This allows the user to skip the Phase I processing and thereby speed up the L2 to L3 conversion time. But this method will ONLY use existing pix0 fit files. It will skip the creation of new ones associated with the current set of L2 files.
3. *unix> bin/DFMS\_PDS\_L2\_to\_L3 -m 600* This method allows the user to process only a specific mode. In the example above the code will only process mode 600 L2 files.
4. In general use the command *unix> bin/DFMS\_PDS\_L2\_to\_L3 -h* for a complete list of command line arguments accepted by the code.

*Note: If you are running CEM files then you must set the DOCEM variable in the file ConfigFile.dat to "true". If running MCP files then set DOCEM to "false".*

2. The code can also be executed via a quasi-batch method which uses an accompanying Perl script *runOneDay.pl*. The Perl script is designed to process CEM and/or MCP data in *natural* blocks. DFMS modes are run in a specific sequence. In between sequences there exists a natural time gap where a new sequence is loaded. The Perl script exploits this time gap to create blocks of L2 files that are related to a specific mode sequence. Typically, there are 2-3 mode sequences per day. The Perl script will look at all CEM and/or MCP L2 files and COPS files in either a *main* directory in called, AllData or in file type specific directories called, MCPData, CEMData, copsBGData, copsNGData, and copsRGData. The user specifies in the input configuration file, *runOneDay.dat*, which method to use and the full pathname to the specific or all-inclusive files directories. The script then defines blocks of L2 files based on the time gap described above and places them (one block at a time) in a directory called *data/OneDayOfData*, which is assigned (in the *ConfigFile.dat* file) to *sL2sourcePath* as defined above. The Perl script will then automatically execute the DFMS code for each block of L2 files. Upon completion of this block of files the script deletes them from the directory *data/OneDayOfData* and then the next block is copied. This loop will continue until all blocks are processed.

The procedure to execute the code using the Perl Script is as follows,



1. Place at least 2 and up to N days of COPS, CEM and/or MCP level 2 data files in the directory called AllData or, alternatively, identify the specific file type directories in the script input configuration file, *runOneDay.dat*.
2. The file ConfigFile.dat must have the following definition for *sL2sourcePath* set,  
 # Location of L2 files relative to env var DFMS\_INSTALL\_DIR  
 sL2sourcePath        data/ OneDayOfData /
3. **unix>** cd \$DFMS\_INSTALL\_DIR
4. To run CEM files use the command,  
**unix>** ./runOneDay.pl CE (making sure that DOCEM is set to "true" in ConfigFile.dat)
5. To run MCP files use the command,  
**unix>** ./runOneDay.pl MC (making sure that DOCEM is set to "false" in ConfigFile.dat)

### Perl Script Configuration file

The perl script, *runOneDay.pl*, requires the user defined input configuration file, *runOneDay.dat* to exist in the same directory as the script. A typical input configuration file looks like the following,

```
# This config file is for the user to setup specific run parameters
# Please make all changes to this file only. Do not edit the perl script

# Full path to Legacy alldata directory
ALLDATA = /Users/marinaldi/xcode/data/AllData/

# Full path to One day of data directory
ONEDAYDIR = /Users/marinaldi/xcode/data/OneDayOfData/

# Full path to MCP L2 file directory
MCPDIR = /Users/marinaldi/xcode/data/MCPData/

# Full path to CEM L2 file directory
CEMDIR = /Users/marinaldi/xcode/data/CEMData/

# Full path to COPS NG file directory
COPSNGDIR = /Users/marinaldi/xcode/data/copsNgData/

# Full path to COPS RG file directory
COPSRGDIR = /Users/marinaldi/xcode/data/copsRgData/

# Full path to COPS BG file directory
COPSBGDIR = /Users/marinaldi/xcode/data/copsBgData/

# Set legacy variable
#   If set to 1 then the code assumes data is coming from a single user created data directory (ALLDATA).
#   If set to 0 then code assumes data is coming from separate data directories, as defined above.
LEGACYPROCESSING = 0

# Set time gap variables
GAPTIMEDELTA = 3540.0 # Gap time in seconds
MAXTIME = 86400.0 # Max Time, 24 hours in seconds
```

## Perl Installation Requirements

The perl script, *runOneDay.pl*, requires the external CPAN packages Date::Calc, Carp::Clan, and Bit::Vector. To install these modules manually proceed as follows,

First download them from CPAN,

For Date::Calc go to: <http://search.cpan.org/~stbey/Date-Calc-6.4/lib/Date/Calc.pod>  
For Carp::Clan go to: <http://search.cpan.org/~kentnl/Carp-Clan-6.06/lib/Carp/Clan.pod>  
For Bit::Vector go to: <http://search.cpan.org/~stbey/Bit-Vector-7.4/Vector.pod>

(Note that these links show the latest versions of each package. Newer ones maybe available)

In each of these CPAN pages look for the download link on the upper right side of the page. Click the download link. For the purpose of this tutorial assume that you have downloaded each of these CPAN external package distributions (Date::Cal, Carp::Clan, Bit::Vector) into the directory, /home/download. For each distribution do the following,

```
unix> cd /home/download
unix> gzip -d Bit-Vector-7.4.tar.gz
unix> tar -xvf Bit-Vector-7.4.tar
unix> cd Bit-Vector-7.4
unix> perl Makefile.pl
unix> make
unix> make test
unix> make install
```

Repeat this process for Carp::Clan and then Date::Calc. Install them in this order.

If necessary then install them to perl so that it knows about them,

```
unix> perl -MCPAN -e shell
cpan> install Bit::Vector
cpan> install Carp::Clan
cpan> install Date::Calc
cpan> exit
```

## The L3 Data PDS Keyword Description


The final data product produced by the software is an archival set of L3 PDS complaint files. The L3 file contains a PDS header followed by 3 Data Object descriptions. These are the DFMS\_HK\_TABLE (DFMS Housekeeping data Table), the DFMS\_MASS\_CAL\_TABLE (DFMS Mass Calibration Table), and the MCP\_DATA\_L3\_TABLE (The L3 mass vs number of ions per spectrum data). Below is a description of the PDS Keywords used in the PDS Header and the DFMS\_HK\_TABLE Object. These are new keywords that are introduced and are unique to the L3 PDS files.

|   |  |
|---|--|
| ROSETTA:ROSINA_CAL_ID4:<br>'None' if no GCU available (after Jan 3. 2015).  | Mass calibration file containing the linear fit for px0 vs mass derived from GCU files.  |
| ROSETTA:ROSINA_CAL_ID4:<br>ROSETTA:ROSINA_CAL_ID6:<br>ROSETTA:ROSINA_PIXEL0_A_MASS<br>ROSETTA:ROSINA_PIXEL0_B_MASS<br>(ROSETTA:ROSINA_INST_MODEL)   | Mass calibration file containing the linear fit for px0 vs mass derived from SLF files.<br>Mass calibration file used, only if a calibration peak was found in that file.<br>Mass at px0 for mass calibration for row A<br>Mass at px0 for mass calibration for row B<br>Instrument model FS (Flight model in Space) or FM (Flight model on Ground)  |
| ROSINA_DFMS_SCI_SIGNAL_CAL_VAL_A<br>ROSINA_DFMS_SCI_SIGNAL_CAL_DEV_A<br>ROSINA_DFMS_SCI_SIGNAL_CAL_VAL_B<br>ROSINA_DFMS_SCI_SIGNAL_CAL_DEV_B<br>ROSINA_DFMS_SCI_OFF_LEVEL_A<br>ROSINA_DFMS_SCI_OFF_STDEV_A<br>ROSINA_DFMS_SCI_OFF_COEFF_FILE<br>ROSINA_DFMS_SCI_OFF_COEFF_C1_A<br>ROSINA_DFMS_SCI_OFF_COEFF_C2_A<br>ROSINA_DFMS_SCI_OFF_COEFF_C3_A<br>ROSINA_DFMS_SCI_OFF_LEVEL_B<br>ROSINA_DFMS_SCI_OFF_STDEV_B<br>ROSINA_DFMS_SCI_OFF_COEFF_C1_B<br>ROSINA_DFMS_SCI_OFF_COEFF_C2_B<br>ROSINA_DFMS_SCI_OFF_COEFF_C3_B<br>ROSINA_DFMS_SCI_GCU_PIXEL0_A<br>ROSINA_DFMS_SCI_GCU_PIXEL0_UNC_A<br>ROSINA_DFMS_SCI_GCU_PIXEL0_B<br>ROSINA_DFMS_SCI_GCU_PIXEL0_UNC_B<br>ROSINA_DFMS_SCI_SELF_PIXEL0_A<br>ROSINA_DFMS_SCI_SELF_PIXEL0_UNCA<br>ROSINA_DFMS_SCI_SELF_PIXEL0_B<br>ROSINA_DFMS_SCI_SELF_PIXEL0_UNCB<br>ROSINA_DFMS_SCI_AVG_PPM_DEV_A | Calibration factor used to convert the L2 data signal into ions/spectra for row A<br>Uncertainty in % of the calibration factor for row A (1%)<br>Calibration factor used to convert the L2 data signal into ions/spectra for row B<br>Uncertainty in % of the calibration factor for row B (1%)<br>offset subtracted for row A (zero order polynomial part)<br>Standard deviation of offset subtracted for row A<br>File used for 3 <sup>rd</sup> order polynomial fit to subtract offset<br>First order coefficient of polynomial fit to subtract offset for row A<br>Second order coefficient of polynomial fit to subtract offset for row B<br>Third order coefficient of polynomial fit to subtract offset for row A<br>Offset subtracted for row B (zero order polynomial part)<br>Standard deviation of offset subtracted for row B<br>First order coefficient of polynomial fit to subtract offset for row B<br>Second order coefficient of polynomial fit to subtract offset for row B<br>Third order coefficient of polynomial fit to subtract offset for row B<br>pix0 (mass calibration) derived from GCU files for row A<br>Uncertainty of px0 (mass calibration) derived from GCU files for row A<br>pix0 (mass calibration) derived from GCU files for row B<br>Uncertainty of px0 (mass calibration) derived from GCU files for row B<br>pix0 (mass calibration) derived from SLF files for row A<br>Uncertainty of px0 (mass cal.) derived from SLF files for row A<br>pix0 (mass calibration) derived from SLF files for row B<br>Uncertainty of px0 (mass calibration) derived from SLF files for row B<br>Deviation of the calculated masses of the peak from their known mass, in parts per million for row A. The deviation of the mass peak is the absolute value of the mass difference. |
| ROSINA_DFMS_SCI_AVG_PPM_DEV_B   | Deviation of the calculated masses of the peak from their known mass, in parts per million for row B. The deviation of the mass peak is the absolute value of the mass difference.   |



DFMS\_INSTALL\_DIR  
/L3hfoLogs,  
QuickLook

## DFMS\_PDS\_L2\_to\_L3 PHASE I Processing Flow

| Directory/Input Files   | Code Algorithm Description and Steps  | Directory/Output Files |
|---|---|------------------------|
|  | <p style="text-align: center;"><b>PHASE I - Process pix0 Fit Data</b></p> <ol style="list-style-type: none"> <li>1 - Gather all available GCU L2 files and store into specific type/mass/resolution array.</li> <li>2 - Gather all available SLF L2 files and store into specific type/mass/resolution array.</li> <li>3 - For each L2 file in a type/mass/resolution array find the specific pix0 value corresponding to the L2 commanded mass m0.</li> <li>4 - Store these m0 vs pix0 pairs into type/mass/resolution/row array.</li> <li>5 - For each type/mass/resolution/row array perform a linear fit to the m0 vs pix0 pairs. There needs to be at least 4 GCU pairs and 3 SLF pairs to perform a fit. The fit function is the simple linear function,               <math display="block">\text{pix0} = a + b * m0</math>               where 'a' is the offset parameter and 'b' is the slope.             </li> <li>6 - Upon a successful linear fit write the resulting fit parameters to x0Fit files.</li> </ol> <p>The two main sources for this Phase I processing are,</p> <p>DFMS_PDS_L2_dir_Class::processPix0FitData()<br/>STEPS 11.1 - 11.10</p> <p>and,</p> <p>DFMS_PDS_L2_dir_Class::findPix0ForGCUSLFType()<br/>STEPS 11.x.0 - 11.x.10</p> |                        |
|   | <p>slfMmHrFiles - Medium Mass / High Resolution SLF files<sup>s</sup><br/>slfHmHrFiles - High Mass / High Resolution SLF files</p>  |                        |

# DFMS\_PDS\_L2\_to\_L3 PHASE II Processing Flow - 1

| Directory/Input Files   | Code Algorithm Description and Steps  | Directory/Output Files                          |
|---|---|---|
| <p style="margin: 0;">ctory:<br/>\$DFMS_INSTALL_DIR<br/>/data/xOFit</p> <p style="margin: 10px 0 0 20px;">→</p> <div style="border: 1px solid blue; background-color: #e6f2ff; padding: 5px; width: fit-content; margin: 10px 0 0 20px;">             Relevant directory:<br/>DFMS_INSTALL_DIR           </div> | <div style="background-color: #e0e0e0; padding: 10px;"> <p style="text-align: center; margin: 0;"><b>PHASE II - Find Peak and Perform Gaussian Fit</b></p> <ol style="list-style-type: none"> <li>1 - Get a list of available X0 Fit files from X0Fit direc.</li> <li>2 - Get the best MPS table and Pixel Gain File</li> <li>3 - Use generic peak finding algorithm to find all peaks in the raw counts data. Determine the peak of interest.</li> <li>4 - Find the L2 offset to the data. This is done by fitting a 3rd order polynomial to the data after all peaks have been removed.</li> <li>5 - Subtract the offset from the raw input L2 data</li> <li>6 - Apply the Overall and pixel Gain correction to the data.</li> <li>7 - Convert the resulting data into lons/pixel.</li> <li>8 - Perform a Gaussian fit to the peak of interest found in step 3 above.</li> </ol> <p>Source code:<br/>DFMS_Convert_L2toL3_Class::processL2() Steps 12.12.1 - 12.12.3.15</p> </div> | <p style="text-align: center; margin: 0;">→</p> |

## Algorithmic Procedure

Below is the step-by-step overview of the setup of the code and then Phase I and II that do the conversion from one PDS level 2 data file to its corresponding PDS level 3 output data file. The step numbers pertain to actual *DFMS\_process\_L2\_to\_L3.cc::main()* source code comment fields. Sub sections refer to either the breakout of the step into sub steps within a loop in the same source code or the sub steps in a referred function or Class method (eg., *DFMS\_PDS\_L2\_dir\_Class::processPix0FitData()*. Refers to the class method *processPix0FitData()* in the Class file *DFMS\_PDS\_L2\_dir\_Class.cc*). These referred steps are indicated in red below.

1. Process Command Line Arguments (CLA).
2. Read configuration file to specify run specific requirements.
3. Read DFMS Constants configuration file to specify run specific DFMS values.
4. Read the latest DFMS mode ID table file.
5. Read the overall gain table file.
6. Retrieve all file information for available Pixel Gain Table files.
  - a. Perform linear interpolation between pixel gain versus time for all pixel gain tables with the same gain step.
7. Retrieve all file information for available GCU MPS table files.
8. Retrieve all file information for available nonGCU MPS table files.
9. Retrieve all file information for available L2 source file data files in the L2 source directory (defined in the configuration file from step 2 above).
10. Retrieve all file information for available COPS (NG/RG/BG) data files.
11. PHASE I processing. This phase finds all GCU and SLF files and performs peak fitting to calculate the required calibration value  $\text{pix}_0$ . For each  $\text{pix}_0$  and commanded mass pair store information for later linear fitting. The process follows the steps below. **Step 11 subsections pertain to *DFMS\_PDS\_L2\_dir\_Class::processPix0FitData()*.**
  - 11.1. Gather all GCU files, if they exist, in the L2 source directory.
  - 11.2. Gather all SLF files in the L2 source directory.
  - 11.3. Find  $\text{pix}_0$  for Low Mass/Low Res GCU files. For each of steps 3-10 (in Phase I processing) the following procedure is enumerated. **Step 11.3 subsections pertain to the steps in the source code *DFMS\_PDS\_L2\_dir\_Class::findPix0ForGCU\_SLF\_Type()*.**
    - 11.3.0. Ascertain date. If date < 1/5/2015 then use GCU data when available. If not, then use SLF for all calibrations.
    - 11.3.1. Get mode ID for this L2 file.
    - 11.3.2. Get the best GCU or nonGCU MPST file from MPST file info.
    - 11.3.3. Get the best Pixel Gain Table file from PGT file info.
    - 11.3.4. Get overall gain for this L2 file.
    - 11.3.5. Read an L2 file. Store PDSheader, HouseKeeping and Data.
    - 11.3.6. Create the  $\text{pix}_0$  fit file object that does all the work.
    - 11.3.7. Process an L2 file and find the comm. mass ( $m_0$ ) and its  $\text{pix}_0$  cal value.
    - 11.3.8. Release dynamically allocated memory.
    - 11.3.9. If there are enough  $m_0/\text{pix}_0$  points perform a linear fit.
    - 11.3.10. Create the X0Fit file for this specific GCU/SLF L2 file type.

- 11.4. Find  $\text{pix}_0$  for High Mass/Low Res GCU files.
  - 11.5. Find  $\text{pix}_0$  for Low Mass/High Res GCU files.
  - 11.6. Find  $\text{pix}_0$  for Medium Mass/High Res GCU files.
  - 11.7. Find  $\text{pix}_0$  for High Mass/High Res GCU files.
  - 11.8. Find  $\text{pix}_0$  for Low Mass/Low Res SLF files.
  - 11.9. Find  $\text{pix}_0$  for High Mass/Low Res SLF files.
  - 11.10. Find  $\text{pix}_0$  for Low Mass/High Res SLF files.
  - 11.11. Find  $\text{pix}_0$  for Medium Mass/High Res SLF files.
  - 11.12. Find  $\text{pix}_0$  for High Mass/High Res SLF files.
12. PHASE II processing. The purpose of this phase is to retrieve the best X0Fit file associated with a specific L2 mode and use it to calculate the calibration value,  $\text{pix}_0$ , needed to create the best mass scale. It then produces the mass scale and writes the resultant PDS L3 file to disk. The process follows the steps below.
- 12.1. Get mode information for the specific L2 file.
  - 12.2. Get the current L2 file name from the L2 files object.
  - 12.3. Create the L3 file info object to hold all L3 information needed by the PDS L3 file.
  - 12.4. Read an L2 file. Store PDSheader, HouseKeeping and Data.
  - 12.5. Check resolution type of this mode. If high res then warn or skip.
  - 12.6. Check file type (ie., MCP,CEM or FAR).
  - 12.7. Get the best GCU or nonGCU MPST file from MPST file info in Step 6.
  - 12.8. Get the best COPS NG/RG/BG file from COPS file info in Step 9.
  - 12.9. Get the best Pixel Gain Table file from PGT file info in Step 6. If 2 or more pixel gain tables exist, for this L2 gain step, then use the pixel gain fit parameters calculated in Step 6a to find the best fit pixel gain values and skip to step 12.11. If 1 or less pixel gain tables exist, for this L2 gain step, then proceed to step 12.10.
  - 12.10. Read best Pixel Gain table for this L2 file.
  - 12.11. Get overall gain for this L2 file.
  - 12.12. Enter main L2 to L3 conversion code. **The next subsections pertain to *DFMS\_ProcessMCP\_Class::processL2()*.**
    - 12.12.1. Find all available  $\text{pix}_0$  fit files.
    - 12.12.2. Get MPS table data from file.
    - 12.12.3. For each of Row A/B perform mass scale calibration. For loop steps
      - ~~12.12.3.1. Optionally write raw data to file for GnuPlot plotting~~
      - 12.12.3.2. Calculate the offset to the raw data.
      - 12.12.3.3. Define peak threshold based on mean offset.
      - 12.12.3.4. Define object to search for all peaks.
      - 12.12.3.5. Get all peaks above threshold.
      - 12.12.3.6. Output peaks found info to screen.
      - 12.12.3.7. Apply offset correction.
      - 12.12.3.8. Apply overall and pixel gain correction.
      - 12.12.3.9. Convert corrected raw count data to number of ions.
      - 12.12.3.10. Perform initialization of peak fitting algorithm.
      - 12.12.3.11. Perform peak fitting using simple Gaussian function.
      - 12.12.3.12. Calculate fit parameter errors.
      - 12.12.3.13. If fit succeeded save fit results for later use.
      - 12.12.3.14. Assign the base level L3 Info values to step 12.3 object.



- 12.12.3.15. If fit did not succeed assign default fit parameters to L3.
- 12.12.3.16. Read latest GCU X0fit file.
- 12.12.3.17. If GCU construct the best mass scale.
- 12.12.3.18. For SLF file read the latest X0fit file.
- 12.12.3.19. If SLF construct the best mass scale.
- 12.12.3.20. If nonGCUonSLF construct the best mass scale.
- 12.12.3.21. ~~Optionally write raw and fit data to file for GnuPlot plotting.~~
- 12.12.3.22. Release dynamically allocated memory for this internal loop.
- 12.12.4. Create mass scale for best calibration.
- 12.12.5. Add COPS NG/RG/BG info to L3 object.
- 12.12.6. Write time series information for special masses to file.
- 12.12.7. Printout results to screen and l3 log file.
- 12.12.8. Get the needed L3 HK data.
- 12.12.9. Form the L3 HK data output array.
- 12.12.10. Form the Calibration data output array.
- 12.12.11. Output GCU MPS table if requested.
- 12.12.12. Define the number of HK values.
- 12.12.13. Create the L3 PDS object.
- 12.12.14. Write the final PDS L3 file to disk.
- 12.12.15. Optionally write L2/L3 data to file for external plotting.
- 12.12.16. ~~Optionally output L3 data for Gnuplot plotting.~~
- 12.12.17. Release dynamically allocated memory used in internal loop.
- 12.13. Release dynamically allocated memory used in L2 to L3 conversion loop.
- 13. Output the number of modes present in this session.
- 14. Output the statistics for the  $PPM_{Diff}$  calculated in this session.
- 15. Calculate and output the total time used for this session.
- 16. Output all end of run statistics to screen, process log, and optionally to alternate L3 information log files.
- 17. Release remaining dynamically allocated memory

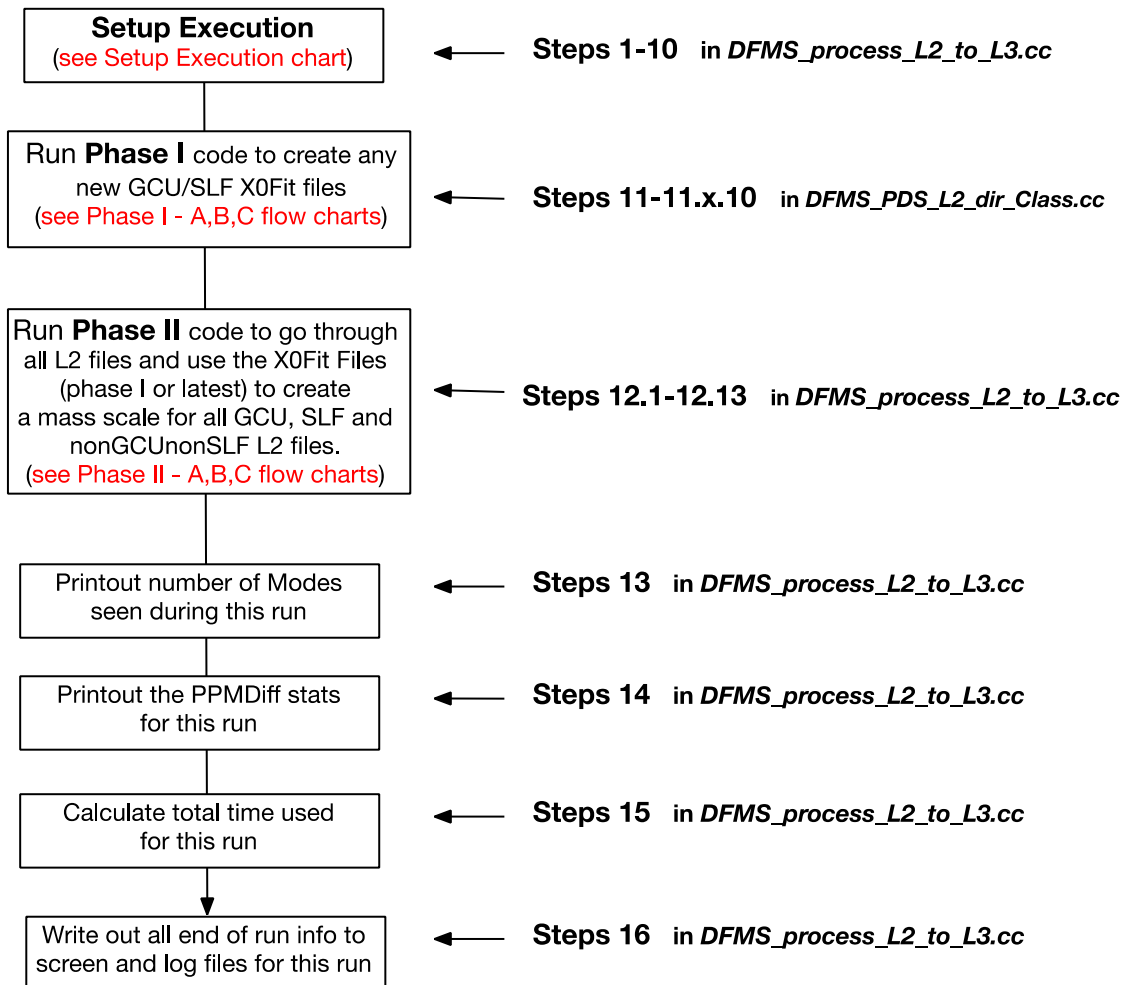
See Appendix 1 for flow chart documents for procedure flow and details.

Note: Although the L2 to L3 conversion is handled one spectra file at a time, the conversion software has been developed to process multiple spectra (all L2 files in a single directory) during the execution of the program. This is desirable not only because of the large amount of data, but because the individual files are tied together as described above, in that, nonGCUonSLF spectra reference recent GCU/SLF spectra for the purposes of mass scale calibration. SLF files also reference recent GCU files for the purpose of mass scale calibration.

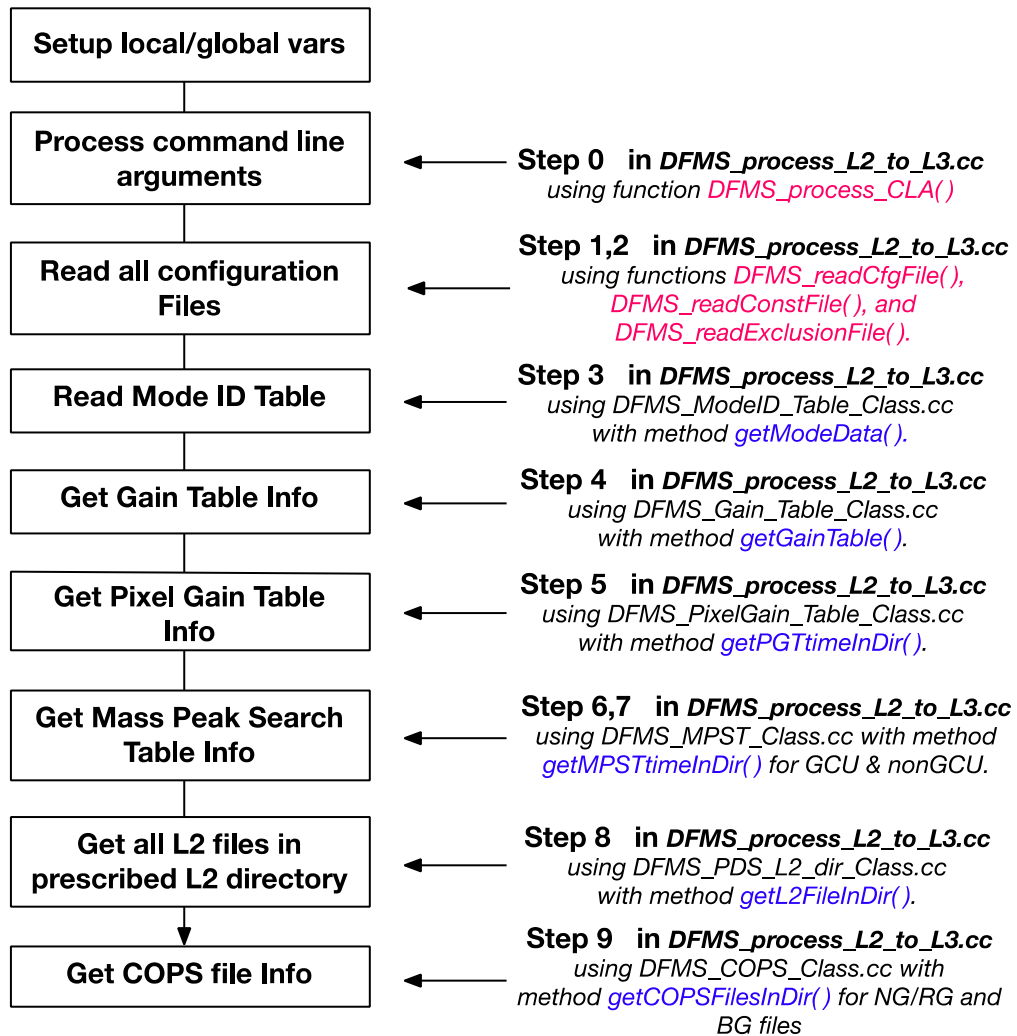
## Appendix 1

### DFMS\_L2\_to\_L3 main() Flow Chart

Step numbers below refer to source code comments

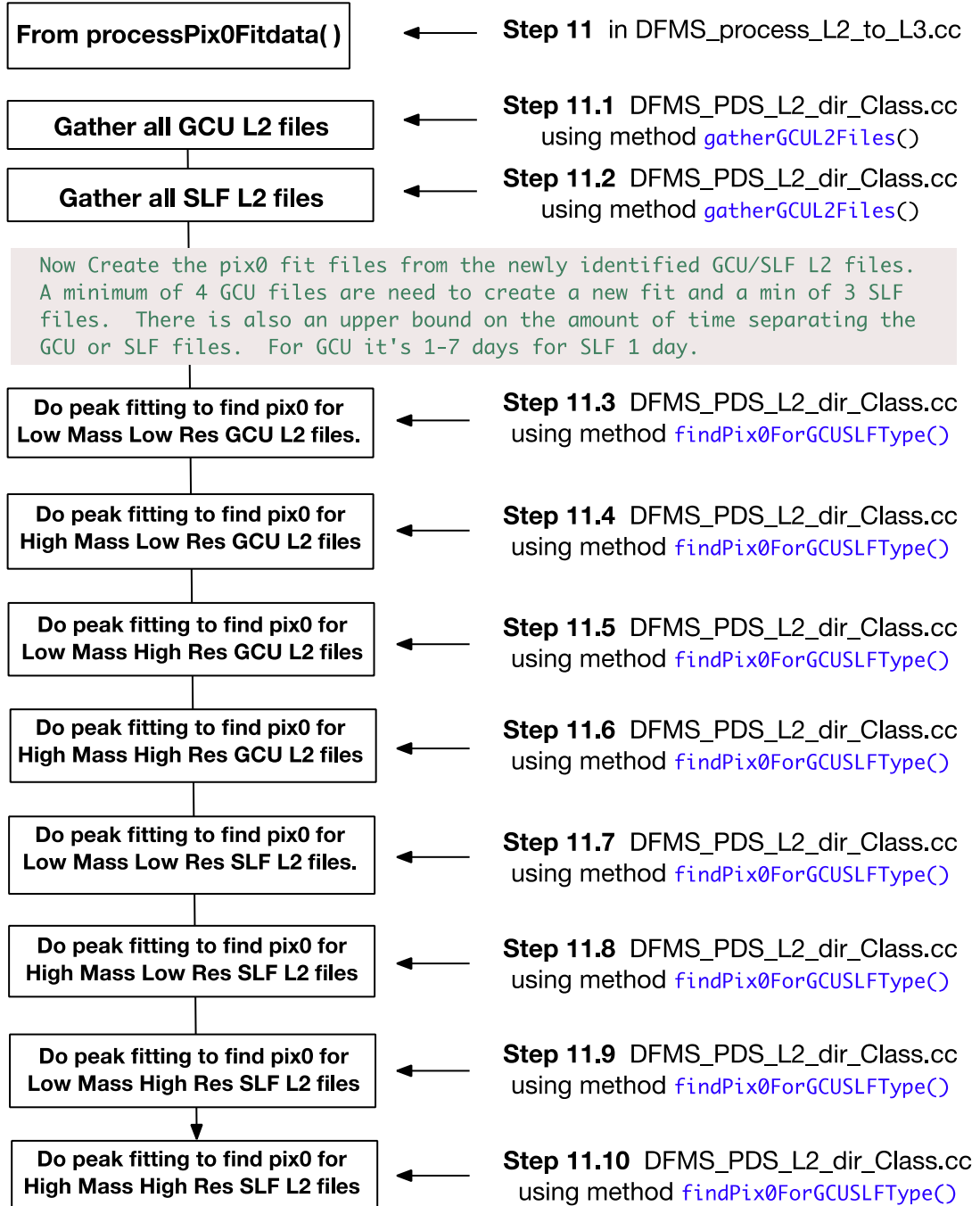


## Setup Execution



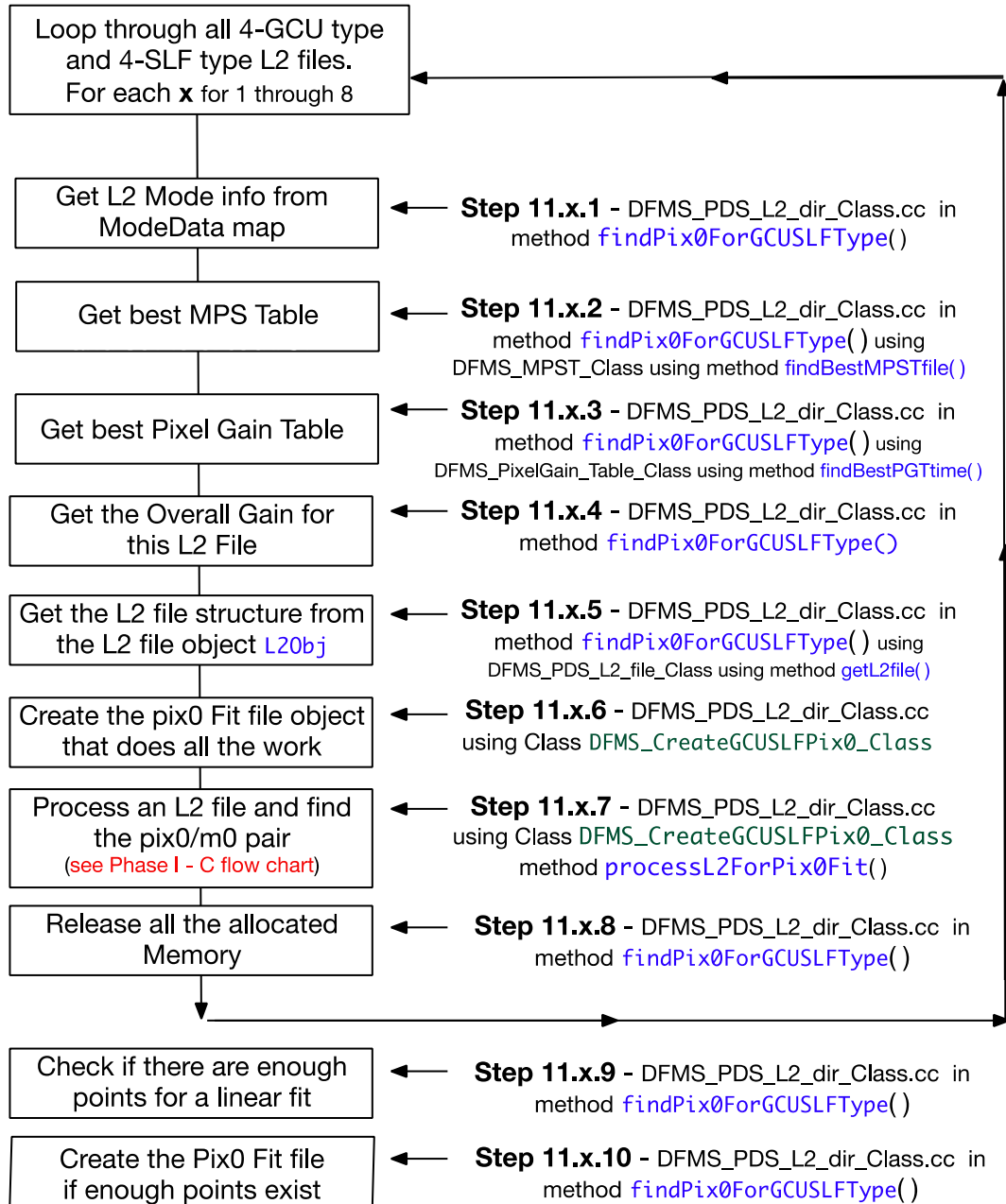
At this point all the information, location, and availability of input files required to convert L2 files into L3 science data files is available in internal variable containers (ie., C++ maps or vectors). Later these containers will be searched for specific files necessary to process one L2 file into an L3 science data file

**Phase I Processing A - Identify GCU/SLF L2 files**  
in DFMS\_PDS\_L2\_dir\_Class.cc using method `processPix0FitData()`



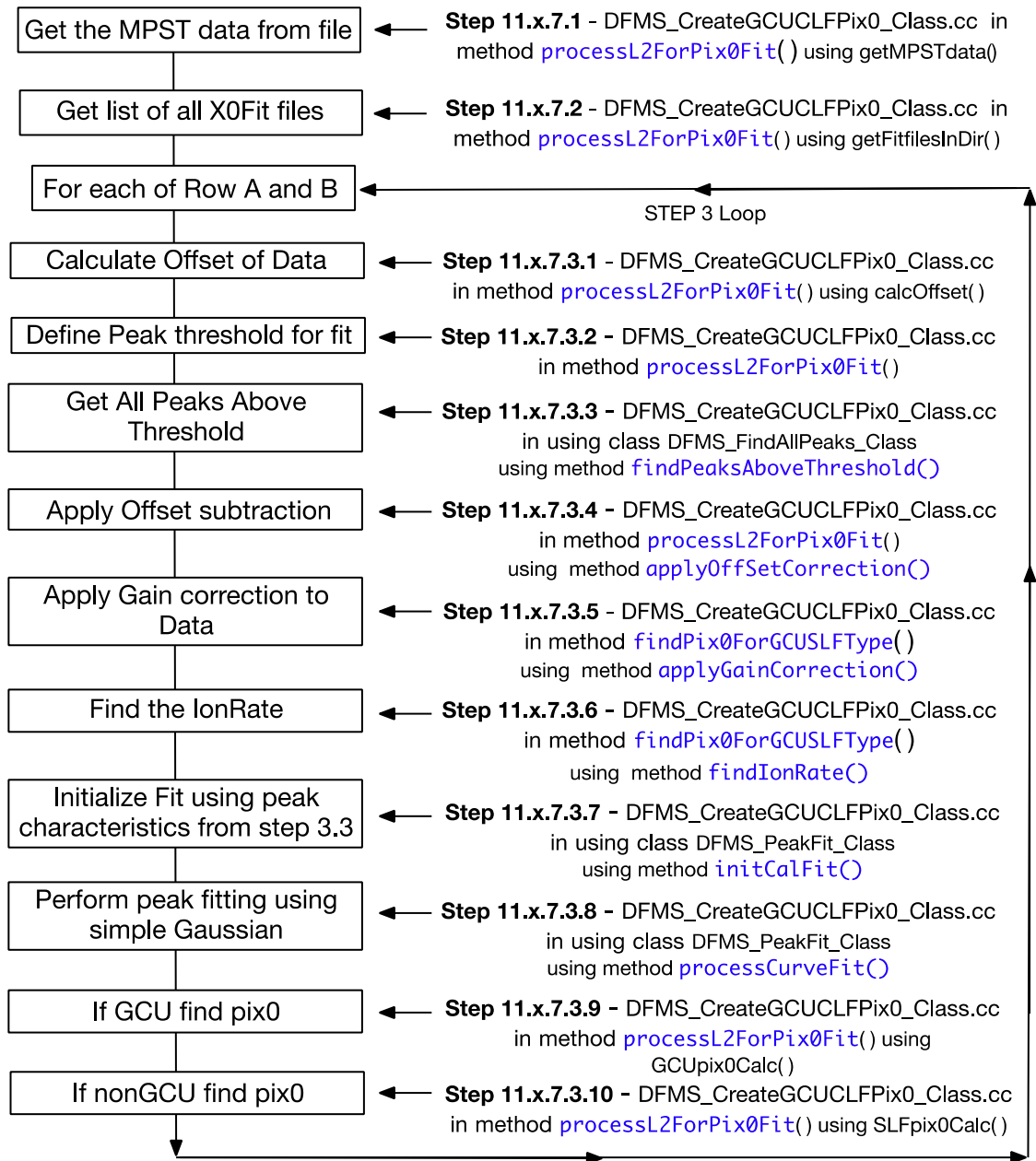
**Phase I Processing B** - Perform Peak Fitting to find pix0 for each commanded mass m0.

using DFMS\_PDS\_L2\_dir\_Class method findPix0ForGCUSLFType()

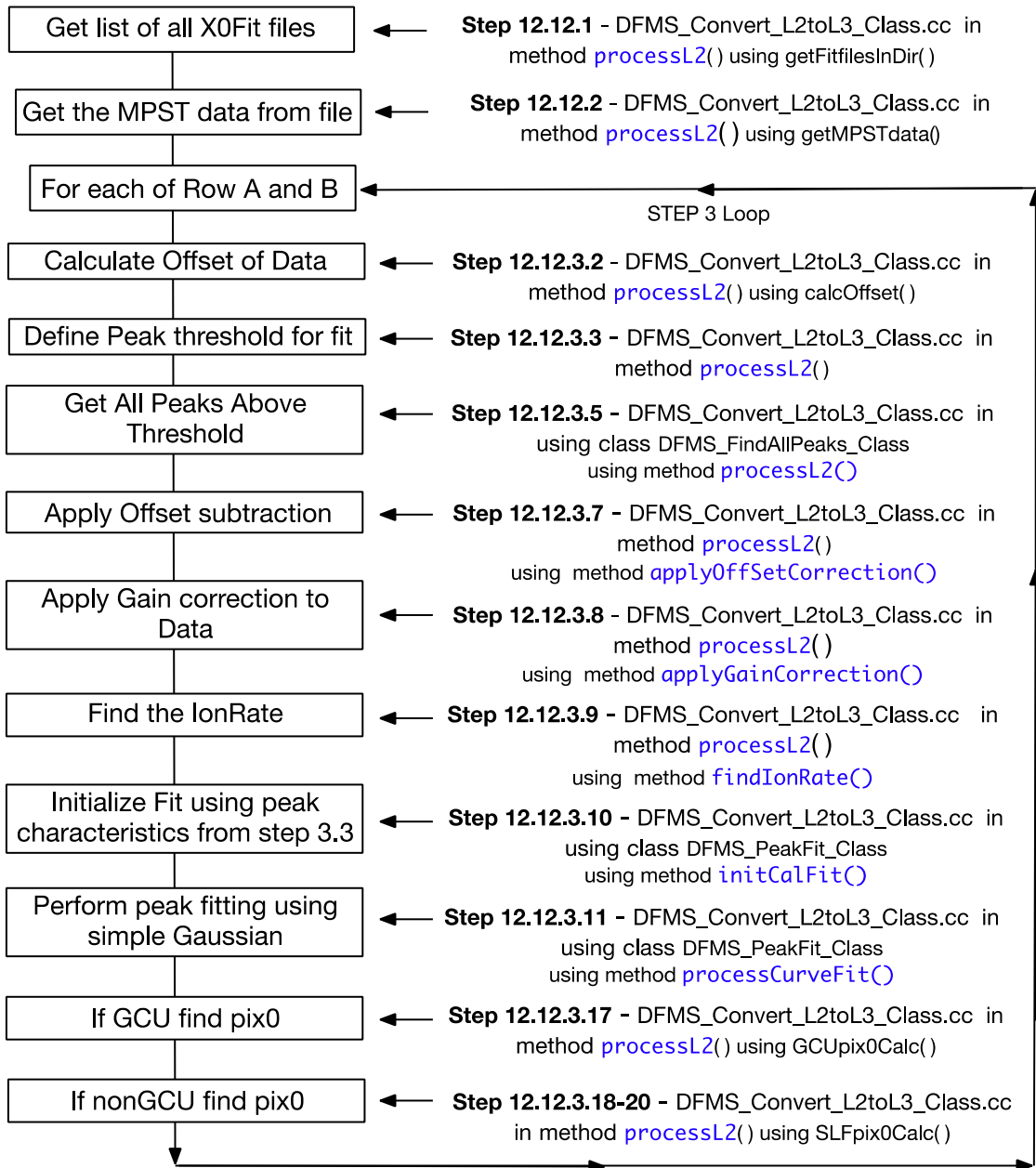


**Phase I Processing C** - Process an L2 File and run the peak fit code to establish pix0 for each commanded mass m0.

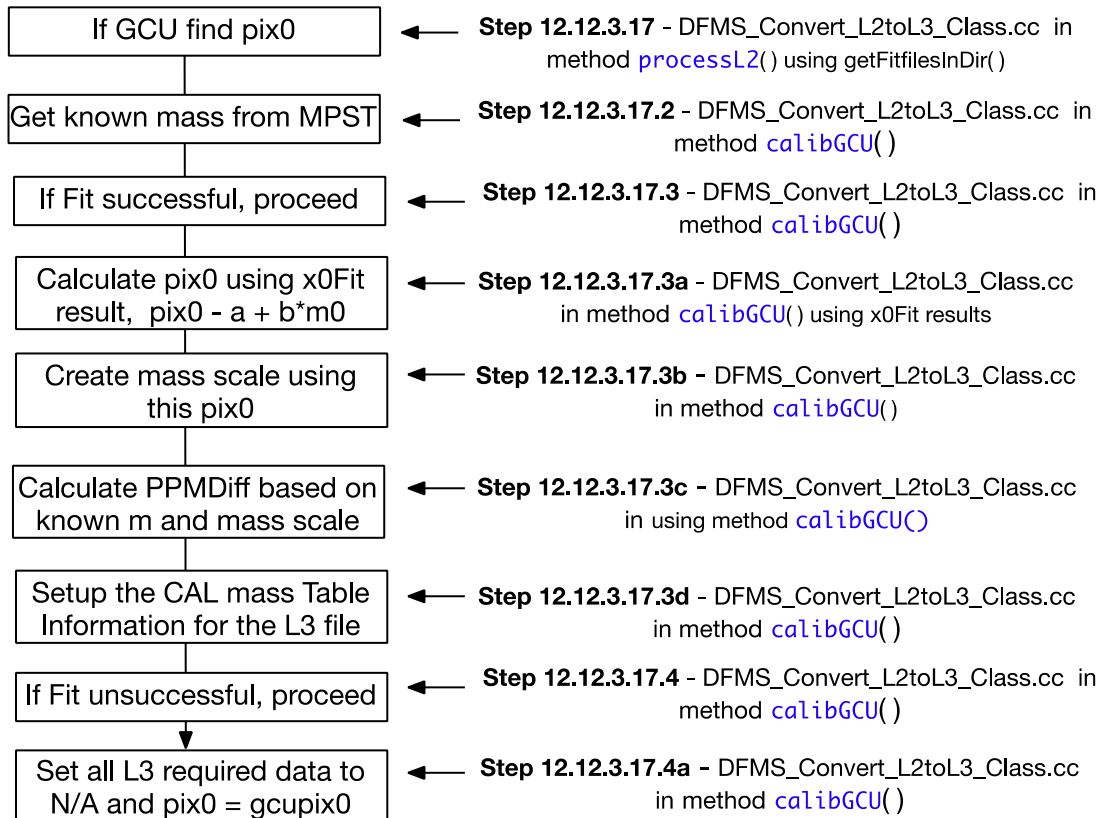
using DFMS\_CreateGCUCLFPix0\_Class method `processL2forPix0Fit()`



**Phase II Processing A** - Process an L2 File and run the peak fit code to establish the best mass scale and ascertain its quality using DFMS\_Convert\_L2toL3\_Class method `processL2()`



**Phase II Processing B** - Show GCU Calibration process  
 using DFMS\_Convert\_L2toL3\_Class method `processL2()` and `calibGCU()`





**Phase II Processing C** - Show SLF Calibration process  
 using DFMS\_Convert\_L2toL3\_Class method `processL2()` and `calibSLF()`

