Interpolation of ASCII-Table Information via Calibration Transfer

F. Masci, 6/18/2003

I. Overview and Aim

One of the options in calibration transfer (caltrans) is to create new calibration products by interpolating two pre-existing calibration products with independent records in the fallback or metadata database tables at the acquisition time of a DCE. The simplest example is interpolation of DN-to-flux conversion factors and their uncertainties. It is envisaged there will be other scalar variables where interpolation may be needed in future. Flexibility in caltrans is therefore required to accommodate this. This document describes a method for standardizing the storage and usage of ASCII table calibration-data under the caltrans interpolation scheme.

II. Method

i. We assume that all possible ASCII calibration data can be stored in IPAC-table format, such as the hypothetical example below. To make it as generic as possible, the column lengths need not be the same. Caltrans need not be concerned with the number of columns or rows—it is just a collection of data whose entries are understood by downstream software (e.g. science pipeline wrappers). The only requirement is that it is in IPAC-table format.

<table>
<thead>
<tr>
<th>Index</th>
<th>fluxconv</th>
<th>errfluxconv</th>
<th>gain</th>
<th>latcoeff</th>
<th>inconst</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>double</td>
<td>double</td>
<td>double</td>
<td>double</td>
<td>real</td>
</tr>
<tr>
<td>1</td>
<td>1.34</td>
<td>0.05</td>
<td>5.34</td>
<td>3.14</td>
<td>1.21</td>
</tr>
<tr>
<td>2</td>
<td>1.54</td>
<td>0.04</td>
<td>4.23</td>
<td>3.21</td>
<td>1.71</td>
</tr>
<tr>
<td>3</td>
<td>1.41</td>
<td>0.03</td>
<td>4.67</td>
<td>3.54</td>
<td>1.31</td>
</tr>
<tr>
<td>4</td>
<td>1.78</td>
<td>0.08</td>
<td>5.21</td>
<td>3.45</td>
<td>1.54</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>3.23</td>
<td>1.25</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>3.11</td>
<td>1.44</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>3.87</td>
<td>1.45</td>
<td></td>
</tr>
</tbody>
</table>

ii. For caltrans to carry out the interpolation rule, at least two instances of a calibration-data table must exist in the database. If there are more than two instances, caltrans should pick the pair of entries which are closest-in-time as it currently does. If there is only one entry, it should “fall-back” to using the closest-in-time entry.

iii. Below is the new proposed input namelist specification to caltrans when interpolation is desired (in this case the range of rule ID’s is 400 and above; rule ID = 0 signifies “fallback”). Command-line equivalents will also exist. There is a new namelist parameter: “CalColName#” which is described below. For simplicity, not all required variable specifications are shown.

```
CalType = 'darkcal; lincal; flatcal; paramtable',
CalOutFname = 'dark.fits; lincal.fits; flat.fits; mipsparam.tbl',
CalRule = '0; 0; 0; 400',
CalColName4 = 'fluxconv; errfluxconv; inconst';
```

iv. The parameter CalColName# specifies a list of column names where interpolation is desired in CalType product number “#” of the CalType list (in this case # = 4). This adds flexibility in specifying more than
one ASCII parameter file in the CalType list where interpolation may or may not be desired. Below is a schematic of the proposed processing flow when CalRule ≥ 400 is specified:

---

**Fig 1. Caltrans processing flow under the interpolation scheme**

- If Rule ID ≥ 400 specified...
  - Check for at least two records in database with appropriate timestamps. Use pair separated closest in time. Fall back to closest-in-time record if only one record.
  - Check to see if both records are ASCII files in IPAC format or FITS files. If the former, proceed with next step, otherwise use FITS interpolation (not discussed here).
  - Search for the “CalColName#” specification input string corresponding to the CalType “#” in list. Abort if it does not exist.
  - Check that all specified CalColName inputs exist in both files and that all column lengths match. If not, abort with appropriate error to stdout.
  - For each column name in the CalColName input string, interpolate values under that column name from each table file at the DCE time.
  - Output product will contain:
    1. Same IPAC-table format as inputs
    2. New interpolated values under all column names specified in the CalColName input string.
    3. For columns where no interpolation was desired, copy entries from closest-in-time product record to output table file.

---

v. The tableio library for reading/writing tables in IPAC format and error checking routines provided therein should be used.

**III. Downstream S/W Requirements**

Once the above functionality is implemented, it is envisaged that all relevant pipeline modules and/or wrappers which make use of the new cal-product format will need updating. In the case of code written in native C or C++, the tableio library can be used. For perl wrappers, it would be advantageous to call a standard perl library for reading IPAC-tables which all systems can use. F. Masci will embark on writing a perl IPAC-table reader for this purpose (which is to be called perltbli).