HIRDLS

SW-HIR-2005

Originator: Charles Cavanaugh

Date: 10 Dec. 2012

Subject/Title:

The Architecture of the HIRDLS Level 2 Preprocessor

Description/Summary/Contents:

The purpose of this document is to create a comprehensive architectural plan of the High Resolution Dynamics Limb Sounder (HIRDLS) Level 2 Preprocessor, hereby known as the L2 Preprocessor, from its already captured requirements. The goal of the plan is to create a system that is, of course, correct, but also to create a system that is easily understood and can be easily designed, developed, extended and maintained. It is assumed that the reader of this document has already read and understood the documented requirements of L2 Preprocessor.

Keywords:

Purpose of this Document:

Oxford University
Atmospheric, Oceanic &
Planetary Physics
Parks Road
OXFORD OXI 3PU
United Kingdom

University of Colorado, Boulder
Center for Limb Atmospheric Sounding
3450 Mitchell Lane, Bldg. FL-0
Boulder, CO 80301

EOS

The Architecture of the HIRDLS Level 2 Preprocessor

Charles Cavanaugh

Table of Contents

List of Figures and Tables

1 Document Purpose and Goal

The purpose of this document is to create a comprehensive architectural plan of the High Resolution Dynamics Limb Sounder (HIRDLS) Level 2 Preprocessor, hereby known as the L2 Preprocessor, from its already captured requirements. The goal of the plan is to create a system that is, of course, correct, but also to create a system that is easily understood and can be easily designed, developed, extended and maintained. It is assumed that the reader of this document has already read and understood the documented requirements of L2 Preprocessor.

2 Architectural Constraints

The L2 Preprocessor architecture must flow freely from the requirements, and must not force design into any particular paradigm. Specifics, such as reusability and portability, must not hinder creativity, and therefore must not be decided by the architecture. Unless listed specifically in the requirements document, development platform and off-the-shelf software use must not be considered by the architectural plan.

The L2 Preprocessor architecture must, however, consider the resource limitations in which the finished software product will exist. The requirements document introduced L2 Preprocessor as a stand-alone, non-graphical, non-embedded scientific application, and as such, resource limitations are ever changing and negotiable with the HIRDLS Program Manager.

3 Architectural Representations

The architecture of L2 Preprocessor will be presented in three different ways: via logical packages, via data flow and via control sequence. The logical package portion will detail how L2 Preprocessor, or one of its packages, is broken into internally cohesive packages. The data flow portion will detail how data flows through L2 Preprocessor or one of its packages. The control sequence portion will detail the order in which packages operate and data flows.

4 L2 Preprocessor

The L2 Preprocessor requirements document outlines what the system must do, which are: ingest a HIRDLS Level 1 Correction (L1C) file, FFT filter and adjust the radiances in the file, spline the data to the HIRDLS altitude grid, and write the filtered, adjusted and splined data to a HIRDLS Level 1R (L1R) file. Figure 1 shows the internal mechanisms, via a combination of the three representations outlined in Section 3, by which L2 Preprocessor fulfills those requirements. The SDP Toolkit, introduced in the L2 Preprocessor Requirements document, is shown in Figure 1 as a collaborator.

4.1 L2 Preprocessor Packages

As shown in Figure 1, L2 Preprocessor consists of as many packages, five, as it has distinct requirements. Those four packages are shown as labeled ovals, and are: Scan Extractor, Radiance Filterer, Scan Transformer, Radiance Adjuster and File Writer. Each of these packages is detailed in later sections of this document.

It is no coincidence that the name of these packages is very similar to their required task. Packages are considered “whats”, and are intended to delineate “what” a system needs to fulfill its requirements, and therefore have noun labels. How a package fulfills its requirements will have verb labels, and “hows” will be detailed in the L2 Preprocessor Design document.

The manner is which the packages are determined, i.e. one package per requirement, is an architecturally purposeful decision, and will be the default mode for all packages within the system, unless strong evidence points otherwise. Also, the manner in which the packages communicate, i.e. via data aggregations, is also a purposeful decision, and will be the default mode unless evidence points otherwise. Lastly, packages must be internally cohesive, and must not allow access to their internal mechanisms to any other package.

Figure 2 shows the dependency hierarchy of the packages within L2 Preprocessor. Data aggregations are architected to have lower dependency, like ductwork in a building. Logical packages are architected to have higher dependency, like utility systems (heating, a/c, etc.) in a building. So in the same manner that heating and air conditioning systems, to be efficient, need ductwork (but not vice-versa), logical packages within a system need data, and are therefore dependent on that data. The dashed arrows indicate dependency direction.

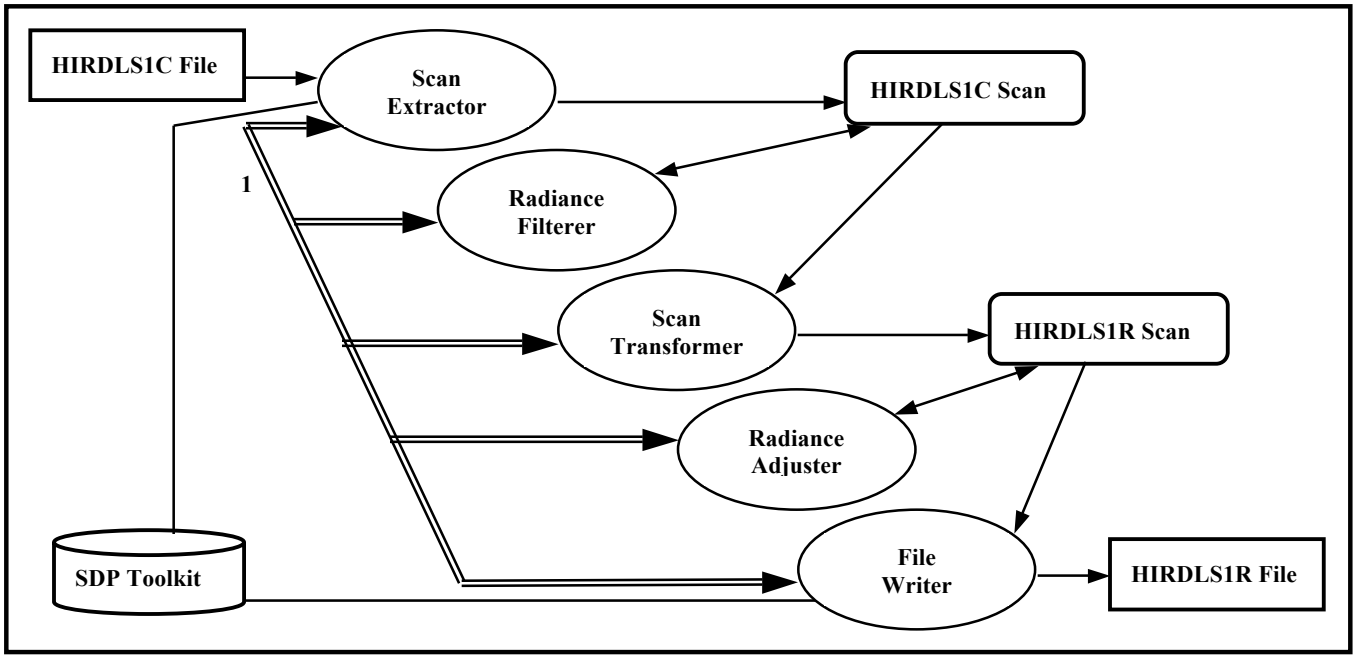

Figure 1 L2 Preprocessor

4.2 L2 Preprocessor Control Sequence

Control sequence within L2 Preprocessor is illustrated in Figure 1 via numbered, double-lined arrows (the numbers indicate sequence). Within L2 Preprocessor, Scan Extractor initiates control by extracting the next scan from the input HIRDLS1C file. The other four packages are then called, in turn, to process that scan. This is repeated until all scans have been extracted from the HIRDLS1C file. This sequence was chosen for two reasons: 1) each package can, potentially, perform its respective task without knowledge of other scans; and 2) to reduce resource usage.

4.3 L2 Preprocessor Data Flow

Beyond the requirements to ingest a HIRDLS1C file and create a HIRDLS1R file, the data flowing within L2 Preprocessor is localized to two data aggregations: HIRDLS1C Scan and HIRDLS1R Scan. As shown in Figure 1 via single-lined arrows (illustrating the direction of data flow), HIRDLS1C Scan is written to by Scan Extractor, read from and written to by Radiance Filterer, and read from by Scan Transformer, while HIRDLS1R Scan is written to by Scan Transformer, read from and written to by Radiance Adjuster, and read from by File Writer.

Given the control sequence outlined in Section 4.2, Scan Extractor is to make available one HIRDLS1C Scan at a time. That doesn't necessarily mean that Scan Extractor ingests one scan at a time from the HIRDLS1C file. That data flow will be described in more detail in Section 5, but from this view, the data flow rate into and out of Scan Extractor looks like "one scan".

Other control sequences outlined in Section 4.2 have Radiance Filterer, Scan Transformer and Radiance Adjuster performing their respective tasks one scan at a time, so at this level, the data flow rate looks like "one scan". The internal view of Scan Transformer could potentially look different, as that package might want to aggregate HIRDLS1C Scans before making HIRDLS1R Scans available. More details on these packages can be found in Sections 6, 7 and 8, respectively.

The last control sequence outlined in Section 4.2 has File Writer, at this level, reading one HIRDLS1R Scan at a time and writing that scan to the HIRDLS1R file. Section 9 details the actual data rates more thoroughly.

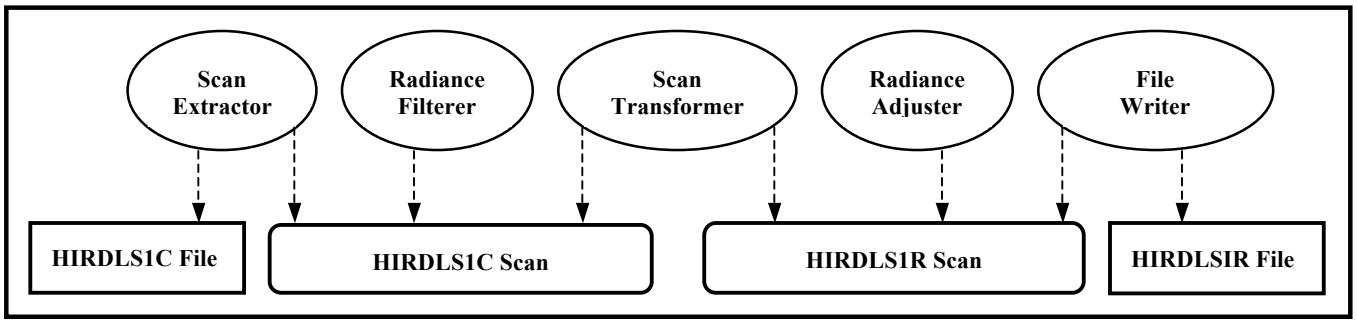

Figure 2 L2 Preprocessor Hierarchy

5 Scan Extractor

The Scan Extractor package, introduced in Figure 1, is tasked to ingest a HIRDLS1C File, and make available the HIRDLS1C Scans found within. Implied in ingestion is to transform the data from "file" units into more identifiable "data" units, since it makes the L2 Preprocessor more maintainable to have the transformation routine(s) isolated to one package. The HIRDLS1C File contents are detailed in the L1 Processor Requirements document, and the data to ingest is implied in Table 1 of the L2 Preprocessor Requirements document. The HIRDLS1C Scan aggregation must, obviously, store all the data necessary for each of the successive packages. The SDP Toolkit will be used to access the HIRDLS1C file.

5.1 Scan Extractor Packages

To accomplish its task, Scan Extractor does not necessarily need further packages, but service packages to encapsulate HIRDLS1C file reading and data transformation would be useful. These specifics will be taken into consideration by the L2 Preprocessor Design document, and used to generate a more detailed approach to the Scan Extractor package.

5.2 Scan Extractor Control Sequence

A valid and efficient control sequence of Scan Extractor will depend on the organization of any auxiliary packages. As stated in Section 5.1, further information will be detailed in the L2 Preprocessor Design document.

5.3 Scan Extractor Data Flow

As noted in Section 4.3, and illustrated in Figure 1, a high-level view of the data flowing into and out of Scan Extractor looks like "one scan". The SDP Toolkit, which will be used to read data from the HIRDLS1C file, works most efficiently when reading a few large "chunks" of data, and bogs down when reading very many small "chunks". Therefore, the data flow rate from HIRDLS1C File into Scan Extractor will be more than just one scan, the exact amount is not known at this time, but will be decided at implementation. Scan Extractor will aggregate the "chunks", and manage the delivery of its output "one scan" rate.

6 Radiance Filterer

The Radiance Filterer package, introduced in Figure 1, is tasked to FFT filter the radiances in a HIRDLS1C Scan. Implied in this task is to read from HIRDLS1C Scan whatever data is necessary to filter the radiances in the scan, and then write the filtered radiances back into the HIRDLS1C Scan. Auxiliary HIRDLS data files will most likely be needed as input to the filtering process for such parameters as channel cut-off frequency, etc. The HIRDLS1C Scan aggregation must store the data needed by Radiance Filterer.

6.1 Radiance Filterer Packages

To accomplish its task, Radiance Filterer does not necessarily need further packages, but service packages to encapsulate auxiliary file reading and math functionality would be useful. These specifics will be taken into consideration by the L2 Preprocessor Design document, and used to generate a more detailed approach to the Radiance Filterer package.

6.2 Radiance Filterer Control Sequence

A valid and efficient control sequence of Radiance Filterer will depend on the organization of any auxiliary packages. As stated in Section 6.1, further information will be detailed in the L2 Preprocessor Design document.

6.3 Radiance Filterer Data Flow

As noted in Section 4.3, and illustrated in Figure 1, a high-level view of the data flowing into and out of Radiance Filterer looks like “one scan”. At this time, it is believed that FFT filtering of a scan’s radiances can happen one scan at a time, and no previous or subsequent scan data is needed, so therefore the data flow into and out of Radiance Filterer is indeed “one scan”.

7 Scan Transformer

The Scan Transformer package, introduced in Figure 1, is tasked to transform a HIRDLS1C Scan into a HIRDLS1R Scan. As outlined in Section 4, the main task of Scan Transformer is to spline the HIRDLS radiances from the HIRDLS1C Scan elevation angle grid to the HIRDLS1R altitude grid. Another important task is the generation of the HIRDLS1R Scan, which is to contain, though perhaps not in its final form, all of one scan’s data that is to be written to the HIRDLS1R File. The L2 Preprocessor Requirements document details the data to be encapsulated in the HIRDLS1R File.

7.1 Scan Transformer Packages

To accomplish its task, Scan Transformer does not necessarily need further packages, but service packages to encapsulate math functionality would be useful. These specifics will be taken into consideration by the L2 Preprocessor Design document, and used to generate a more detailed approach to the Scan Transformer package.

7.2 Scan Transformer Control Sequence

A valid and efficient control sequence of Scan Transformer will depend on the organization of any auxiliary packages. As stated in Section 7.1, further information will be detailed in the L2 Preprocessor Design document.

7.3 Scan Transformer Data Flow

As noted in Section 4.3, and illustrated in Figure 1, a high-level view of the data flowing into and out of Scan Transformer looks like “one scan”. At this time, it is believed that scan transformation can happen one scan at a time, and no previous or subsequent scan data is needed, so therefore the data flow into and out of Scan Transformer is indeed “one scan”.

8 Radiance Adjuster

The Radiance Adjuster package, introduced in Figure 1, is tasked to adjust the radiances in a HIRDLS1R Scan. Implied in this task is to read from HIRDLS1R Scan whatever data is necessary to adjust the radiances in the scan, and then write the adjusted radiances back into the HIRDLS1R Scan. Auxiliary HIRDLS data files will most likely be needed as input to the adjusting process for such parameters as channel weights, etc. The HIRDLS1R Scan aggregation must store the data needed by Radiance Adjuster.

Radiance Adjuster works on HIRDLS1R Scans because, at this time, radiance adjustment happens in altitude space, which is the y-axis unit in HIRDLS1R Scans. It is possible that this adjustment approach could change to elevation angle space, which is the y-axis unit in HIRDLS1C Scans, and therefore Radiance Adjuster could potentially be moved between Radiance Filterer and Scan Transformer.

8.1 Radiance Adjuster Packages

To accomplish its task, Radiance Adjuster does not necessarily need further packages, but service packages to encapsulate auxiliary file reading and math functionality would be useful. These specifics will be taken into consideration by the L2 Preprocessor Design document, and used to generate a more detailed approach to the Radiance Adjuster package.

8.2 Radiance Adjuster Control Sequence

A valid and efficient control sequence of Radiance Adjuster will depend on the organization of any auxiliary packages. As stated in Section 8.1, further information will be detailed in the L2 Preprocessor Design document.

8.3 Radiance Adjuster Data Flow

As noted in Section 4.3, and illustrated in Figure 1, a high-level view of the data flowing into and out of Radiance Adjuster looks like “one scan”. At this time, it is believed that adjusting a scan’s radiances can happen one scan at a time, and no previous or subsequent scan data is needed, so therefore the data flow into and out of Radiance Adjuster is indeed “one scan”.

9 File Writer

The File Writer package, introduced in Figure 1, is tasked to write the filtered, adjusted and splined data in a HIRDLS1R Scan to a HIRDLS1R File. To accomplish this task, this package must be able to read data from the HIRDLS1R Scan aggregation, and create and write the data to a HIRDLS1R File, using the SDP Toolkit. The HIRDLS1R File contents are detailed in the L2 Preprocessor Requirements document.

9.1 File Writer Packages

To accomplish its task, File Writer does not necessarily need further packages, but service packages to encapsulate HIRDLS1R file access would be useful. These specifics will be taken into consideration by the L2 Preprocessor Design document, and used to generate a more detailed approach to the File Writer package.

9.2 File Writer Control Sequence

A valid and efficient control sequence of File Writer will depend on the organization of any auxiliary packages. As stated in Section 9.1, further information will be detailed in the L2 Preprocessor Design document.

9.3 File Writer Data Flow

As noted in Section 4.3, and illustrated in Figure 1, a high-level view of the data flowing into and out of File Writer looks like "one scan". As mentioned in Section 5.3, the SDP Toolkit most efficiently reads and writes files in "chunks". Therefore, the data flow rate from File Writer to HIRDLS1R File will be more than just one scan, the exact amount is not known at this time, but will be decided at implementation. The data flow rate into File Writer will still be "one scan".

10 Diagnostics

Though not explicitly detailed in this document, a diagnostics repository must be a part of L2 Preprocessor. The diagnostics package must be available to all packages within L2 Preprocessor, and must be able to store diagnostic information and write that information to a file. The exact diagnostics to write, and the exact form of the output file, will be determined at implementation.