

Originator: Charles Cavanaugh

Date: 10 Dec. 2012

Subject/Title: **The Architecture of the HIRDLS
Level 1 Correction Processor**

Description/Summary/Contents:

The purpose of this document is to create a comprehensive architectural plan of the High Resolution Dynamics Limb Sounder (HIRDLS) Level 1 (L1) Correction Processor, hereby known as LIC Processor, 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 LIC Processor.

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 1 Correction Processor

Charles Cavanaugh

Table of Contents

Table of Contents	.i
List of Figures and Tables	.ii
Section 1 Document Purpose and Goal	.1
Section 2 Architectural Constraints	.1
Section 3 Architectural Representations	.1
Section 4 L1C Processor	.1
Section 4.1 L1C Processor Packages	.1
Section 4.2 L1C Processor Control Sequence	.2
Section 4.3 L1C Processor Data Flow	.2
Section 5 File Copier	.3
Section 6 X Corrector	.3
Section 6.1 X Corrector Packages	.3
Section 6.2 X Corrector Control Sequence	.3
Section 6.3 X Corrector Data Flow	.4
Section 7 Scan Extractor	.4
Section 7.1 Scan Extractor Packages	.4
Section 7.2 Scan Extractor Control Sequence	.4
Section 7.3 Scan Extractor Data Flow	.5
Section 8 Scan Corrector	.5
Section 8.1 Scan Corrector Packages	.5
Section 8.2 Scan Corrector Control Sequence	.5
Section 8.3 Scan Corrector Data Flow	.5
Section 9 Scan Writer	.5
Section 9.1 Scan Writer Packages	.6
Section 9.2 Scan Writer Control Sequence	.6
Section 9.3 Scan Writer Data Flow	.6
Section 10 Diagnostics	.6

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 1 (L1) Correction Processor, hereby known as L1C Processor, 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 L1C Processor.

2 Architectural Constraints

The L1C Processor 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 L1C Processor architecture must, however, consider the resource limitations in which the finished software product will exist. The requirements document introduced L1C Processor 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 L1C Processor will be presented in three different ways: via logical packages, via data flow and via control sequence. The logical package portion will detail how L1C Processor, or one of its internal packages, is broken into internally cohesive packages. The data flow portion will detail how data flows through L1C Processor or one of its packages. The control sequence portion will detail the order in which packages operate and data flows.

4 L1C Processor

The L1C Processor requirements document outlines what the system must do, which are ingest a HIRDLS Level 1 (HIRDLS1) File, correct the contaminated radiances within, and write the corrected radiances (and other ancillary data) to a HIRDLS1C File. Figure 1 shows the internal mechanisms, via a combination of the three representations outlined in Section 3, by which L1C Processor fulfills those requirements. The SDP Toolkit, introduced in the L1C Processor Requirements document, is shown in Figure 1 as a collaborator.

4.1 L1C Processor Packages

As shown in Figure 1, L1C Processor consists of five packages, which do not necessarily align with its three requirements. The rationale behind this is twofold: 1) L1C Processor works on only a small portion of the data within HIRDLS1 File, and therefore reading all of the file would be highly inefficient; and 2) there are four components of the correction, and they have to be decoupled from each other. Those five packages are shown as labeled ovals, and are: File Copier, Oscillation Corrector, Emissions Corrector, Obscuration Corrector and Error Corrector. 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 L1C Processor Design document.

Figure 2 shows the dependency hierarchy of the packages within L1C Processor. 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.

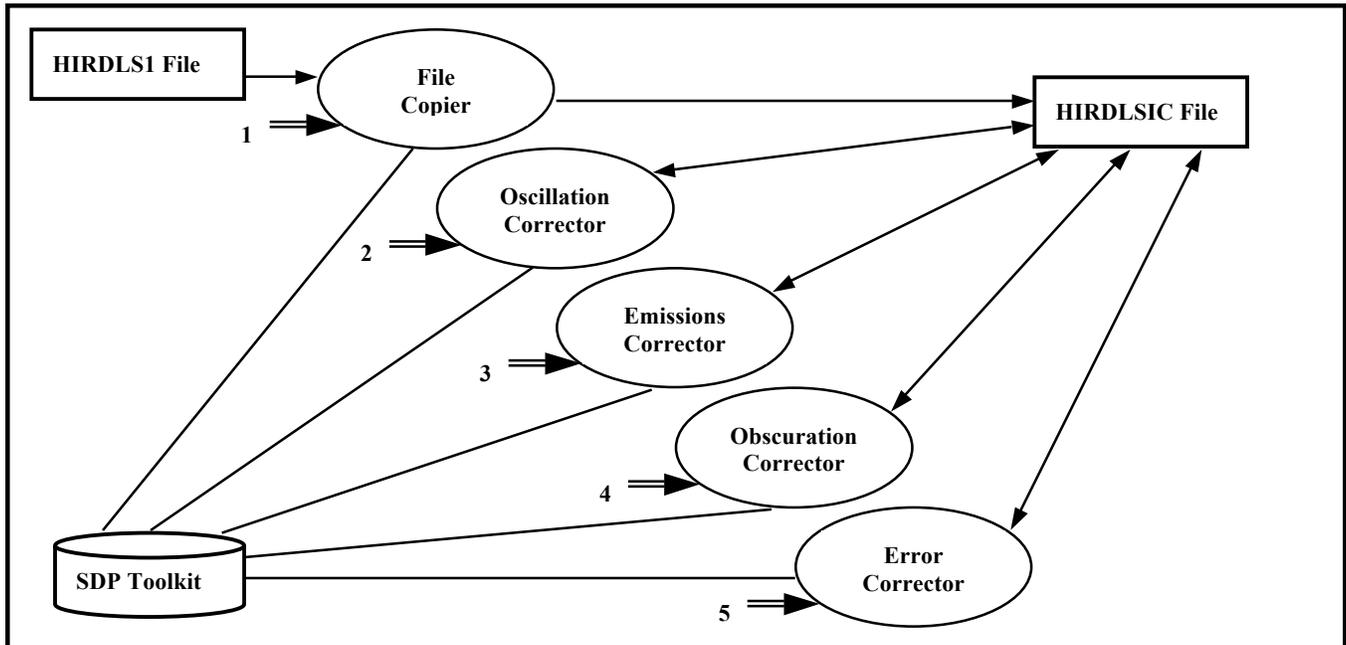


Figure 1 L1C Processor

4.2 L1C Processor Control Sequence

Control sequence within L1C Processor is illustrated in Figure 1 via numbered, double-lined arrows (the numbers indicate sequence). Within L1C Processor, File Copier initiates control, as indicated by the “1”. After that, considering the requirement to keep the corrections decoupled, the sequence chosen is to have the different corrections not communicate with each other via a data aggregation, but instead perform their respective corrections, in order, on the file itself. Of the correctors, Oscillation Corrector must act first (as indicated by the “2”), as it is necessary to remove the oscillating signal before attempting emissions or obscuration correction. Emissions Corrector (“3”) and Obscuration Corrector (“4”) then act, in order. The last correction is Error Corrector (“5”). This is the standard operational control sequence. It is assumed that this sequence could be differently ordered, or missing a correction, in non-operational, experimental mode. It is also assumed there will be very many iterations of each of the four corrections, as what the HIRDLS team is attempting has no precedence, and lessons learned will flow into the next iteration.

4.3 L1C Processor Data Flow

Beyond the requirements to ingest a HIRDLS1C file, and create a HIRDLS1R file, no additional data flows between the packages of L1C Processor. This is a purposeful decision, as outlined in Sections 4.1 and 4.2. As shown in Figure 1 via single-lined arrows (illustrating the direction of data flow), File Copier ingests the HIRDLS1C File and creates the HIRDLS1R File. All correction packages read from and write to HIRDLS1R File.

Given the control sequence outlined in Section 4.2, all data flows within L1C Processor look like “all”. Internally, though, these data rates (which are also shown in Figure 1 via single-lined arrows) will most likely be more “bursty”, given what we know about how the SDP Toolkit works, and therefore will be detailed more thoroughly in subsequent sections.

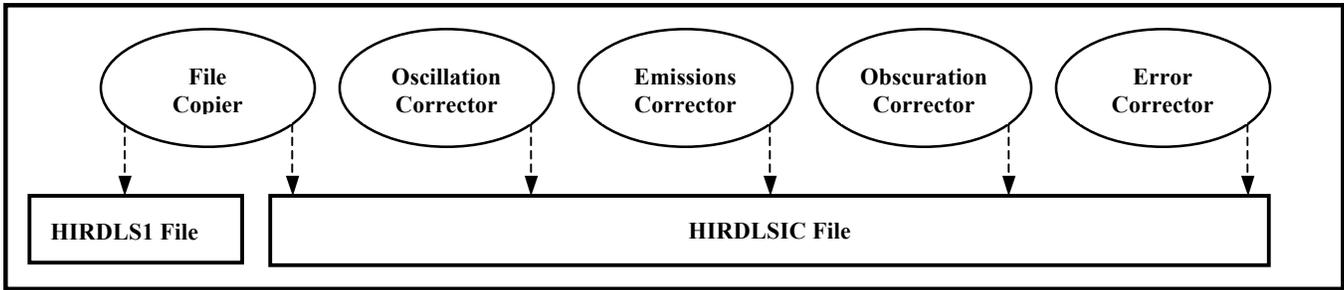


Figure 2 L1C Processor Hierarchy

5 File Copier

The File Copier package, introduced in Figure 1, is tasked to create a copy of HIRDLS1 File into HIRDLS1C File. This is necessary because HIRDLS1 File and HIRDLS1C File represent two of the HIRDLS project’s standard data deliveries, and obviously HIRDLS1 File, as input to L1C Processor, must be read-only. With so many file copy utilities available, it doesn’t make sense to create new software to accomplish File Copier’s task, unless implied in file copying is changing something fundamental about the creation of HIRDLS1C File. At this time, nothing is implied, and therefore this architecture document relegates the task of File Copier to an already existing software package, to be determined at platform decision time.

6 X Corrector

The four corrector packages, Oscillation Corrector, Emissions Corrector, Obscuration Corrector and Error Corrector, all have fundamentally similar tasks, i.e. read the radiances and/or errors from the input/output HIRDLS1C File, perform the respective correction, and write the corrected radiances and/or errors back into the HIRDLS1C File. Therefore it is assumed, at this time, that their respective architectures will be so similar as to preclude the need for four separate architecture sections. This section will refer to the correctors as “X Corrector”, and the “X” will be a placeholder for any given corrector.

6.1 X Corrector Packages

As shown in Figure 3, X Corrector consists of three packages, one for each task. These three packages are shown as labeled ovals, and are: Scan Extractor, Scan Corrector and Scan Writer. Scan Extractor is used to extract HIRDLS scans from HIRDLS1C File. Scan Corrector reads radiances and/or radiance errors from HIRDLS1C Scan, applies the appropriate correction, and then writes the radiances and/or radiance errors back into HIRDLS1C Scan. Scan Writer reads HIRDLS1C Scan and writes the corrected radiances and/or radiance errors back into HIRDLS1C File. These three packages will be discussed in Sections 7, 8 and 9.

6.2 X Corrector Control Sequence

Control sequence within X Corrector is illustrated in Figure 3 via numbered, double-lined arrows. Scan Extractor initiates control (denoted with “1”), sequentially extracting HIRDLS1C Scans and passing them to Scan Corrector (“2”), which corrects the scan and then passes them to Scan Writer (“3”). This is the default control sequence. It might be that a corrector needs “spin up” data, and might need two passes through the data. If that is the case, sequence “1” and “2” (and potentially “3”) might run twice, with Scan Corrector creating an aggregation of spin-up data. It’s not known at this time if this is the case with any of the correctors.

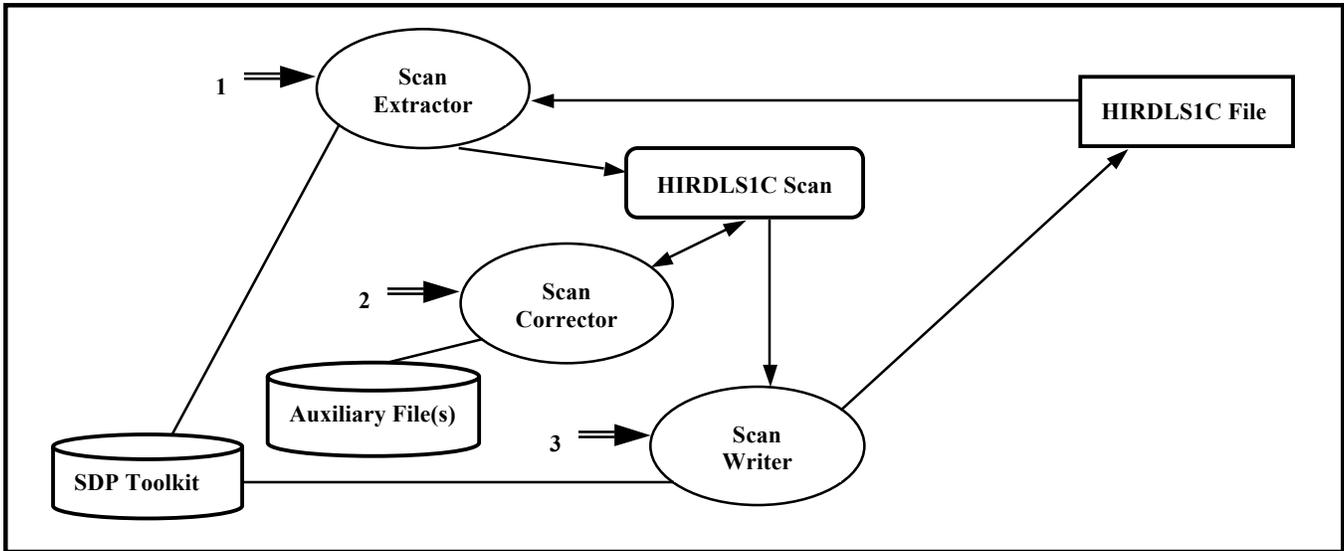


Figure 3 X Corrector

6.3 X Corrector Data Flow

Beyond the requirements to read from and write to a HIRDLIS1C file, the data flowing within X Corrector is localized to one data aggregation: HIRDLIS1C Scan. As shown in Figure 3 via single-lined arrows (illustrating the direction of data flow), HIRDLIS1C Scan is created by Scan Extractor, read from and written to by Scan Corrector, and read from by Scan Writer.

Given the control sequence outlined in Section 6.2, data flow into and out of HIRDLIS1C Scan, at this level, looks like “one scan”, but given SDP Toolkit optimization and corrector considerations, the rate is probably more “chunky”, and will be detailed in Sections 7, 8 and 9.

7 Scan Extractor

The Scan Extractor package, introduced in Figure 3, is tasked to ingest a HIRDLIS1C File, and make available the HIRDLIS1C Scans found within. Implied in ingestion is to transform the data from “file” units into more identifiable “data” units, since it makes the LIC Processor more maintainable to have the transformation routine(s) isolated to one package. The HIRDLIS1C File contents are detailed in the LIC Processor Requirements document. The HIRDLIS1C Scan aggregation must, obviously, store all the data necessary for the given corrector to perform its task. As mentioned in Section 6, SDP Toolkit will be used to access the HIRDLIS1C file.

7.1 Scan Extractor Packages

To accomplish its task, Scan Extractor does not necessarily need further package, but service packages to encapsulate HIRDLIS1C file reading and data transformation would be useful. Further information will be detailed in the LIC Processor Design document.

7.2 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 7.1, further information will be detailed in the LIC Processor Design document.

7.3 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.

8 Scan Corrector

The Scan Corrector package, introduced in Figure 3, is tasked to apply a given component of the radiance anomaly correction to the radiances and/or the radiance errors. Implied in this task is to read from HIRDLS1C Scan whatever data is necessary to correct the data in the scan, and then write the corrected data back into the HIRDLS1C Scan. Auxiliary HIRDLS data files will most likely be needed as input to the given correction process. The HIRDLS1C Scan aggregation must store the data needed by Scan Corrector.

8.1 Scan Corrector Packages

To accomplish its task, Scan Corrector 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 LIC Processor Design document, and used to generate a more detailed approach to the Scan Corrector package.

8.2 Scan Corrector Control Sequence

A valid and efficient control sequence of Scan Corrector will depend on the organization of any auxiliary packages. As stated in Section 8.1, further information will be detailed in the LIC Processor Design document.

8.3 Scan Corrector Data Flow

As noted in Section 6.3, and illustrated in Figure 3, a high-level view of the data flowing into and out of Scan Corrector looks like “one scan”. It is not yet known if any of the correctors will need an aggregate of scans to perform their appropriate correction, though the assumption is that none will need an aggregation. Given this, the data flow rate out of Scan Corrector is one scan, that being the same scan that immediately flowed into Scan Corrector.

9 Scan Writer

The Scan Writer package, introduced in Figure 3, is tasked to write the corrected radiances and/or radiance errors from HIRDLS1C Scan back into HIRDLS1C File. This package is, in all practical purposes, the inverse of Scan Extractor, and therefore must have the same knowledge of HIRDLS1C Scan and HIRDLS1C File as Scan Extractor. This architectural document does not specifically call for these two packages to be unified, but the LIC Processor Design document might create a design that includes a “Scan Accessor” package, that includes an “Extract” and “Write” contract. This would encapsulate all HIRDLS1C File access, including any data transformations.

9.1 Scan Writer Packages

To accomplish its task, Scan Writer does not necessarily need further packages, but service packages to encapsulate HIRDLS1C File writing and data transformations would be useful. These specifics will be taken into consideration by the LIC Processor Design document, and used to generate a more detailed approach to the Scan Writer package.

9.2 Scan Writer Control Sequence

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

9.3 Scan Writer Data Flow

As noted in Section 6.3, and illustrated in Figure 3, a high-level view of the data flowing into and out of Scan Writer looks like “one scan”. Scan Writer, like Scan Corrector, is presented one HIRDLS1C Scan at a time, so the data flow rate into Scan Writer is one scan. The data flow rate out of Scan Writer, though, is not one scan, but a “chunk” of data. Like Scan Extractor, Scan Writer uses SDP Toolkit to perform file I/O, so the same “chunk” optimization applies. And like Scan Extractor, the size of the chunk of data written to HIRDLS1C File will be determined at implementation.

10 Diagnostics

Though not explicitly detailed in this document, a diagnostics repository must be a part of LIC Processor. The diagnostics package must be available to all packages within LIC Processor, 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.