HIRDLS

SW-HIR-2010

Originator: Charles Cavanaugh

Date: 10 Dec. 2012

Subject/Title:

The User's Guide of the HIRDLS Level 1 Correction Processor

Description/Summary/Contents:

The purpose of this document is to describe the building, running and validating of the High Resolution Dynamics Limb Sounder (HIRDLS) Level 1 Correction Processing System. The goal of this document is to detail the description sufficiently, such that the user requires no more guidance than that what is listed in this document.

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 User's Guide of the HIRDLS Level 1 Correction Processor

Charles Cavanaugh

Table of Contents

List of Figures and Tables

1 Document Purpose and Goal

The purpose of this document is to describe the building, running and validating of the High Resolution Dynamics Limb Sounder (HIRDLS) Level 1 Correction Processing System. The goal of this document is to detail the description sufficiently, such that the user requires no more guidance than that what is listed in this document.

2 References

The HIRDLS Level 1 Correction Processing System, hereby known as L1C Processor, has had its requirements, architecture and design already detailed in previous documents. It is not necessary to read those documents before using L1C Processor, as those documents detail the construction of the system. However, it would be helpful to read those documents before using L1C Processor, as those documents also contain useful information on system input and output files, and also contain information on the purpose of the system.

3 Creating the System

L1C Processor is included in the DAP (Delivered Algorithm Package) as a gzipped file, called "L1CProcessor.gz". This section will detail how to create the system.

3.1 Extracting the System

To build L1C Processor, copy the gzipped L1C Processor file to the directory in which you want the system to be built, and then extract the files (thereby building the system) by using the Unix "tar" utility. This will create a directory that includes all the source code, make files, and control files that you will need. However, an input HIRDLS standard product file (HIRDLS1) will be necessary to run L1C Processor, and that is described in Section 5.3. Figure 1 shows a practical example of the steps in this section.

```
hal:/hal/cavanaug/L1CProcessor> ls
-rw-r--r-- 1 cavanaug hirdls 99757807 Sep 24 11:11 L1CProcessor.gz
hal:/hal/cavanaug/L1CProcessor> tar xf L1CProcessor.gz
hal:/hal/cavanaug/L1CProcessor> ls
-rw-r--r-- 1 cavanaug hirdls 99757807 Sep 24 11:11 L1CProcessor.gz
drwxr-xr-x 4 cavanaug hirdls      140 Apr  4 15:47 v7.11.0/
```

Figure 1 Extracted System

3.2 Platform Dependencies

L1C Processor, overall, was developed and implemented to be Unix platform independent and ANSI compatible. L1C Processor uses ASCII files and HDF5 files during processing, and both of these file structures are platform independent.

3.3 External Dependencies

In order to run, L1C Processor needs access to the SDP Toolkit. L1C Processor uses this toolkit to perform such tasks as time conversion and control file reading. As of this document version, the current toolkit version is 5.2.18. L1C Processor should be built with this version, as it is not guaranteed that L1C Processor will work correctly with previous versions. Also, the SDP Toolkit libraries that L1C Processor accesses must be compiled with the same compiler as used on L1C Processor

(see Section 3.4). The SDP Toolkit also provides environment-creating scripts that must be run before compiling or running L1C Processor. These scripts are platform-dependent, and can be found in the toolkit directory, in the appropriate platform "bin" subdirectory. Figure 2 shows a listing that could be used, depending on the platform.

```
/usr/local/TOOLKIT5.2.18gnu/bin/linux/pgs-env.ksh
/usr/local/TOOLKIT5.2.18gnu/bin/linux/pgs-dev-env.ksh
/usr/local/TOOLKIT5.2.18gnu/hdfeos/bin/linux/hdfeos_env.ksh
```

Figure 2 External Scripts

3.4 Compiling the System

As detailed in its design document, L1C Processor is a collection of four different correction sub-processors, and therefore consists of four different executables. Luckily, the compilation process is the same for all four sub-processors (L1C-W, L1C-K, L1C-A, L1C-E). This section will use the second of the four sub-processors, L1C-K, to describe the compilation process, but know that the only difference is the sub-processor subdirectory and moniker. If you are compiling L1C Processor for the first time, please repeat the below steps, substituting out "L1C-K" for "L1C-W", "L1C-A" and "L1C-E", in turn.

To create the L1C Processor L1C-K sub-processor executable, go to the "driver" subdirectory of the "L1C-K/C++/packages" subdirectory of the build you created. In that subdirectory are 5 Makefiles that control compilation of the L1C-K sub-processor. At most, only two of them, "Makefile.exec" and "Makefile.global", should need editing.

"Makefile.global" contains compilation flag definitions, and these can be platform-specific. The default listings of "CFLAGS" and "EXECCFLAGS" are minimal and should work on all platforms, but you are certainly welcome to add your favorite compilation flags. The default for "FASTCFLAGS" turns optimization on, to level 2. Again, this should work for most cases. The defaults for "WARNCFLAGS" and "DEBUGCFLAGS" should also work for most cases. This file also contains a soft reference to the compiler to use to build the L1C-K sub-processor. The default is GNU's g++ compiler.

"Makefile.exec" contains the library listing to use for compilation. If these libraries are different on your machine, you will need to adjust this listing. And like "Makefile.global", "Makefile.exec" contains the same compiler soft link.

"Makefile.system" should not need editing, but contains the three included compilation targets: "systemfast", "systemdebug" and "systemwarn". These targets control which compilation flags are used during compilation. The default way to build the L1C-K sub-processor is to use the "systemfast" target. If you want to be able to access the L1C-K sub-processor with a debugger during a run, use the "systemdebug" target. If you want to see compilation warnings during compile, use the "systemwarn" target. It is recommended that for any extensive run jobs, you use the "systemfast" target to generate the L1C-K sub-processor. To actually build the L1C-K sub-processor, invoke the make command with one of these three targets, i.e., at the Unix prompt, enter "make systemfast" (without the parentheses). Upon successful completion, the sub-processor executable, called "L1C-K", will have been created in the "L1C-K/run/sandbox" subdirectory, as shown in Figure 3. Please note that the listing shown in Figure 3 is specific to the L1C-K sub-processor. The listing will look different for the other three sub-processors, but will contain the appropriate executable once these steps are re-followed for that sub-processor.

3.5 Concatenating the System

When you have compiled versions for all four sub-processors, you will need to concatenate them into one processor, L1C Processor. Go to L1C Processor's root directory, the creation of which is detailed in section 3.1. Once there, enter "build.ksh" at the Unix prompt (without the parentheses). This will create a "run" sub-directory, which will contain all the executables and input files that are necessary to run L1C Processor. This example assumes that you are operating under a Korn shell. If not, use the appropriate execution command for your shell. Figure 4 shows the directory listing after the build.ksh script has been executed successfully.

```
hal:/hal/cavanaug/L1CProcessor/v7.11.0/L1C-K/run> ls
-rwxr--r-- 1 cavanaug hirdls 3263516 Oct 30 09:54 L1C-K*
-rw-r--r-- 1 cavanaug hirdls    9731 Oct 30 09:54 L1C-K_history
-rw-r--r-- 1 cavanaug hirdls   26046 Oct 30 09:54 L1C-K_pcf.txt
-rw-r--r-- 1 cavanaug hirdls     523 Oct 30 09:54 L1C-K_usfmask
-rw-r--r-- 1 cavanaug hirdls  329776 Oct 30 09:54 RS16C.he5
-rw-r--r-- 1 cavanaug hirdls  329776 Oct 30 09:54 RS17C.he5
-rw-r--r-- 1 cavanaug hirdls  329776 Oct 30 09:54 RS18C.he5
-rw-r--r-- 1 cavanaug hirdls     379 Oct 30 09:54 SpaceElevations.txt
-rw-r--r-- 1 cavanaug hirdls     365 Oct 30 09:54 TranslationAngles.txt
```

Figure 3 L1C-K Run Directory Listing

```
hal:/hal/cavanaug/L1CProcessor/v7.11.0> ls
-rwxr--r-- 1 cavanaug hirdls   443 Oct 30 09:52 build.ksh*
-rw-r--r-- 1 cavanaug hirdls  7919 Oct 30 09:52 HIRDLS1C.mcf
drwxr-xr-x 4 cavanaug hirdls    26 Oct 30 09:58 L1C-A/
drwxr-xr-x 4 cavanaug hirdls    26 Oct 30 09:58 L1C-E/
drwxr-xr-x 4 cavanaug hirdls    26 Oct 30 09:58 L1C-K/
-rwxr-xr-x 1 cavanaug hirdls 11031 Oct 30 09:52 L1C.py*
drwxr-xr-x 4 cavanaug hirdls    26 Oct 30 09:58 L1C-W/
-rw-r--r-- 1 cavanaug hirdls 12061 Oct 30 09:52 proc.py
drwxrwxr-x 3 cavanaug hirdls    34 Oct 30 09:59 run/
-rw-r--r-- 1 cavanaug hirdls   382 Oct 30 09:52 usfmask
```

Figure 4 Root Directory Listing

4 Editing the System

It is strongly recommended that you successfully complete Section 3 before you do any editing, to make sure that any issues that might arise are not due to faulty instructions in this user's guide. If you edit the source code, you will of course need to recompile the system. Please note that you only need to recompile the sub-processor(s) that you change. If you edit any of the input files in the DAP, there is no need to recompile the system, unless of course you change the internal structure of a file. Any changes to the system should be documented in the appropriate "history" ancillary file.

5 Running the System

To run L1C Processor, it is assumed you have followed the steps in Section 3 or Section 4, for all four sub-processors, and have a compiled version of the system. Congratulations! That was the difficult step. This section will detail how to run the system

5.1 Building the Run

To build the run, you will first need to create a directory in which you want the run to happen. As detailed in the next section, you will need to consider how much free space you have in that directory. Once you have that directory established or newly created, copy all the files from the "run/sandbox" subdirectory into the run directory. Also copy the "usfmask" file from the "run" subdirectory (one level up from the "sandbox" subdirectory).

5.2 Resource Dependencies

Running L1 Processor requires at least 2 Gbytes of available memory and up to 800 Mbytes of available storage space (depending on the extent of the processing request, and assuming the input HIRDLS1 file is referenced via a symbolic link). L1C Processor requires only one thread.

5.3 Ancillary Files

Not included in L1C Processor's build is the input HIRDLS L1 file necessary for a successful run. The specific file necessary is dependent on the processing request, and a DAP that included all HIRDLS L1 files would be impossibly wieldy. HIRDLS generates the L1 file as part of its standard processing run. If you are running L1C Processor only, and not also L1 Processor first, you will need to acquire the input L1 file from ESDIS. For a standard day's run, you will need only the L1 file for that respective day. Section 5.4 details how L1C Processor is notified of the input L1 file.

5.4 Invoking the System

L1C Processor is invoked by calling the python script "L1C.py" with the input "usfmask" text file. This input text file, an example of which is shown in Figure 5, contains parameters read in by L1C Processor. These parameters are then used to fill in the run file needed by the SDP Toolkit. The value of the parameter "L1CVERSION" in the "Processor Specifics" section should be edited, if necessary, to reflect the current version. In the "Input Files" section, only the value of the parameter "HIRDLS1", which specifies the input HIRDLS L1 file, would need editing, as this value changes with each day's run. All the values for the parameters in the "Output Files" section can be changed to whatever you like, with the "HIRDLS1C" parameter value reflecting the name of the HIRDLS1C file that L1C Processor will create. The values of the parameters in the "Processing Time Range" section should be ignored, as L1C Processor does not currently use them. To reiterate, once the "usfmask" file contains valid values, invoke L1C Processor by entering, at the Unix prompt, "L1C.py usfmask" (without the parentheses).

```
# usfmask
# Mask file used by run script to fill L1X process control file
# Charles Cavanaugh

# Processor Specifics
L1CVERSION = v7.11.0

# Input Files
HIRDLS1CMCF = HIRDLS1C.mcf
HIRDLS1     = %HIRDLS1%

# Output Files
HIRDLS1C  = %HIRDLS1C%
L1CREPORT = L1CReport.txt

# Processing time range
PROCESSINGSTARTTIME = ${data_start_time}
PROCESSINGSTOPTIME  = ${data_end_time}
```

Figure 5 Usfmask Parameter Value Listing

6 Validating the System

To validate the system, it is assumed you have followed the steps in Section 5, and have an output HIRDLS1C file. Congratulations! You are 2/3 the way to having a useful HIRDLS Level 1C file. This section will detail how to validate the system.

6.1 Toolkit Log Files

In the directory in which L1C Processor was run, you should have 3 ASCII toolkit log files. These are named "LogReport", "LogStatus" and "LogUser". The "LogStatus" file is the useful one. This is the file into which SDP Toolkit writes its errors, warnings and reports. There is an endless permutation of these three types of messages, and it is impossible to detail them all, but if the only message is the ubiquitous "W A R N I N G" message about Toolkit version mismatch, then L1C Processor has run successfully, in the view of SDP Toolkit.

6.2 System Log Files

Upon finishing, L1C Processor generates an ASCII file called "L1CReport.txt", which is a status report of the L1C Processor run. L1C Processor was designed to always create this file, no matter the status of the run. The only time this file won't be generated is if L1C Processor dumps core. If this is the case, you need to go back to Section 4 and review why this happened.

Figure 6 details the formatted listing of "L1CReport.txt", which are occurences that L1C Processor deems worthy of special note. Figure 6 shows a typical, non-eventful run of L1 Processor. Of special note is the last line. If the last line does not start with "Normal Termination", then L1C Processor did not normally terminate. If this is the case, the message string after the colon will give a succinct reason why L1C Processor terminated abnormally, and this can be used as a starting to point to determine what went wrong.

Upon examination of this status report file, it will be obvious that each of the four sub-processors has its own section, and each sub-processors section has exactly the same occurrences reported. What could be different is the number of occurrences.

The first occurrence line, for each sub-processor, is the number of scans that were identified. The next four lines count the occurrences of identified scans that had issues, i.e. radiometric contamination of the field of view, invalid azimuth angles, invalid altitude values during elevation scans, and issues with the sub-processor's respective correction focus. The sixth occurrence line, number of scans corrected, will be the number of scans identified minus the sum of the scans that had issues.

Each sub-processor also has its own termination status line. If a sub-processor terminates normally, the status line will contain the string "Normal Termination", and the next sub-processor will be executed. However, if a sub-processor terminates abnormally, the status line will contain "Abnormal Termination" and a terse explanation of the abnormal termination, and no subsequent sub-processors will be executed. Restated: if any part of L1C Processor terminates abnormally, the last line in the status report file will contain the string "Abnormal Termination".

6.3 Metadata

A normally-terminated run of L1C Processor will generate a metadata file for the output HIRDLS1C file, and the name of the file will be the same as the HIRDLS1C file, but with ".met" attached at the end of the name. The information in this file is used for ingestion of the HIRDLS1C file into the ESDIS system, and does not contain much useful information. The parameters that include "DATE" and/or "TIME" could be examined to check veracity of the run.

6.4 Graphical & System Tools

There are numerous graphical and/or system tools that will closely examine the fields inside the HIRDLS1C file. Included in this list are "h5ls" and "h5dump". The HIRDLS program relied heavily on Matlab and IDL programs to examine the contents of all the generated HIRDLS data products. These are not included in the DAP, as these were generated with minimal documentation and minimal notation, and were intended for in-house use only. If you are interested in using any of these, please contact your HIRDLS representative.

```
Report file for L1C-W
       4867  Scans identified
          0  Scans with radiometric contamination
          0  Scans with invalid azimuth
          0  Scans with invalid elevations
          0  Scans with wave correction failure
       4867  Scans corrected
L1C-W Normal Termination : No anomalies during processing
Report file for L1C-K
       4867  Scans identified
          0  Scans with radiometric contamination
          0  Scans with invalid azimuth
          0  Scans with invalid elevations
          0  Scans with kapton correction failure
       4867  Scans corrected
L1C-K Normal Termination : No anomalies during processing
Report file for L1C-A
       4867  Scans identified
          0  Scans with radiometric contamination
          0  Scans with invalid azimuth
          0  Scans with invalid elevations
          0  Scans with area correction failure
       4867  Scans corrected
L1C-A Normal Termination : No anomalies during processing
Report file for L1C-E
       4867  Scans identified
          0  Scans with radiometric contamination
          0  Scans with invalid azimuth
          0  Scans with invalid elevations
          0  Scans with error correction failure
       4867  Scans corrected
L1C-E Normal Termination : No anomalies during processing
```

Figure 6 L1CReport.txt Listing